HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

DIPLOMARBEIT

Herr
Maik Eulitz

Entwicklung von Handlungsanweisungen zur Umsetzung von responsive Websites unter Nutzung von verfügbaren Standardmechanismen bei Content Management Systemen.

2015

Fakultät **Elektro- und Informationstechnik**

DIPLOMARBEIT

Entwicklung von Handlungsanweisungen zur Umsetzung von responsive Websites unter Nutzung von verfügbaren Standardmechanismen bei Content Management Systemen.

Autor:
Maik Eulitz

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
eit09wI-D

Erstprüfer:
Prof. Dr. rer. pol. Dirk Pawlaszczyk

Zweitprüfer:
Dipl.Ing.(FH) Martin Rettke

Mittweida, 01 2015

Bibliografische Angaben

Eulitz, Maik: Entwicklung von Handlungsanweisungen zur Umsetzung von responsive Websites unter Nutzung von verfügbaren Standardmechanismen bei Content Management Systemen., 66 Seiten, 41 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Elektro- und Informationstechnik

Diplomarbeit, 2015

Referat

Es folgt eine Analyse von Content Management Systemen bei der Entwicklung von responsive Webseiten. Bei der Analyse wird ein vorgegebener Prototyp in verschiedene Content Management Systeme implementiert. Dies geschieht unter Verwendung aller gegebenen Standardmechanismen der Systeme. Anschließend erfolgt eine Bewertung der Implementierung und eine abschließende Aufstellung von allgemeinen Handlungsanweisungen für die Erstellung von responsive Webseiten in Content Management Systemen.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Abkürzungsverzeichnis

AEM ..... Adobe Experience Manager, Seite 14

CMS ..... Content Management System, Seite 1

CQ ..... Communique, Seite 14

CRXDE ..... Content Repository Extreme Development Enviroment, Seite 26

CSS ..... Cascading Style Sheets, Seite 6

GmbH ..... Gesellschaft mit beschränkter Haftung, Seite 12

HTML ..... Hypertext Markup Language, Seite 6

JSP ..... JavaServer Pages, Seite 26

pt ..... Point, Seite 8

px ..... Pixel, Seite 3

UI ..... User Interface, dt.: Benutzeroberfläche, Seite 26

IV. Vorwort

Die folgende Diplomarbeit bildet den Abschluss meiner Studienzeit und das Ende eines erlebnisreichen und mit vielen Erfahrungen bestückten Abschnittes meines Lebens. Die Erstellung dieser Diplomarbeit erfolgt in Partnerschaft mit der Content Management Abteilung der T-Systems Multimedia Solutions GmbH und der Hochschule Mittweida.

Das Thema „Entwicklung von Handlungsanweisungen zur Umsetzung von responsive Websites unter Nutzung von verfügbaren Standardmechanismen bei Content Management Systemen“ gab mir noch einmal die Möglichkeit das erworbene Wissen in der Hochschule und der Firma anzuwenden. Mir hat die Erstellung dieser Arbeit viel Freude bereitet. Des Weiteren gab mir diese Arbeit einen anderen Blickwinkel auf das Thema Content Management Systeme.

Eine Danksagung ist an dieser Stelle angebracht. Ich möchte insbesondere meinen beiden Betreuern, Martin Rettke und Prof. Dr. Dirk Pawlaszczyk, für die vielen Anregungen und die unkomplizierte Abwicklung dieser Arbeit danken.

Weiterhin danke ich dem Applikation Management Team bei der moralischen und fachlichen Unterstützung, vor allem Sebastian Heckmann, Benjamin Stelzner und Martin Rettke. Ein Extra-Dankeschön gilt Antje Hofmann für die Idee zum Thema und die enorme Unterstützung und Anregungen während der Erstellung der Arbeit. Auch Matthias Redmann möchte ich für seine fachliche Hilfe bei schwierigen Fragen in Sachen Layout und Frontendentwicklung danken.

Besonderer Dank richtet sich an meine Lektoren, die viel Arbeit in das Korrekturlesen gesteckt haben und mir sowohl sprachlich als auch inhaltlich weiterhelfen konnten. Ein großer Dank geht auch an meine Freunde, die mir die diplom- und unifreie Zeit sehr angenehm gestaltet haben.

Zum guten Schluss kommen die wichtigsten Personen, welche mich während des gesamten Studiums unterstützt und ermutigt haben und wenn es schwierig wurde, auch wussten wie sich mich motivieren können. Danke Esther, Mama, Papa, Matthias, Peggy, Sybil, Matthias, Ursula und Hektor.

1 Einleitung

1.1 Erwartung

Das stetige Wachstum des World Wide Web und das Bedürfnis sowie die Notwendigkeit sich in diesem zu präsentieren, lässt immer neue und erweiterte Systeme zum Erstellen, Verwalten und Präsentieren von Inhalten auf dem Markt erscheinen. Im Zuge dieser Diplomarbeit sollen drei ausgewählte Content Management Systeme (CMS) auf ihre Fähigkeiten geprüft werden, mit Hilfe der Standardmechanismen eine prototypisch, sich responsive verhaltende, Beispielseite nachzubauen.

Anhand dieses Vorgehens soll eine Vergleichbarkeit der drei Systeme gewährleistet werden und die besonderen Features der einzelnen Systeme genauer analysiert werden.

1.2 Thesen

Anhand der Recherche- und Analyseergebnisse sowie der Erkenntnisse aus der Konzeption und Umsetzung der prototypischen Beispielseite in den drei verschiedenen CM Systemen werden die folgenden Thesen diskutiert.

1. ***Responsive Design ermöglicht ein analoges Verhalten auf verschieden Endgeräten***
   Der Nutzer sollte eine Seite auf allen Endgeräten wiedererkennen können. Lediglich eine Anpassungen von Schriftgrößen und Anordnungen sollten durchgeführt werden. Die grundlegende Struktur und der Aufbau sollten auch auf mobilen Geräten zu finden sein. Die Struktur wird mit Hilfe von „div“ Elementen als ein Container-Modell gebaut.
2. ***Responsive Design unterstützt die Barrierefreiheit.***
   Responsive Webseiten unterstützen die Barrierefreiheit, das bedeutet es ermöglicht Nutzern mit einer körperlichen Einschränkung, ebenso wie unerfahren Nutzern, diese uneingeschränkt und ohne Hilfe Dritter nutzen zu können.
3. ***Responsive Design Webseiten sind immer inhaltlich gekürzt.***
   Aufgrund der kleineren Anzeigemöglichkeit müssen die Texte und Inhalte auf den mobilen Geräten gekürzt und in einer abgespeckten Form dargestellt werden.
4. ***Alle Content Management Systeme brauchen ein responsive Design Feature.***
   Das Thema responsive Design ist zur Zeit in einer starken Entwicklungsphase, daher sollten auch Content Management Systeme den Nutzern die Möglichkeit geben, ihre Seiten responsive darstellen zu lassen.

5. ***Nicht jedes Content Management System besitzt solch ein Feature.***
   Nicht alle Content Management Systeme geben den Nutzern eine Unterstützung zur Entwicklung von responsive Webseiten an die Hand, da die Möglichkeit besteht, Webseiten ohne zusätzliches Feature in einem Content Management System responsive darzustellen.
6. ***Content Management Systeme sind für die Erstellung von responsive Design geeignet.***
   Es ist möglich responsive Webseiten mit jedem aktuellen Content Management System zu erstellen.
7. ***Allgemeine Handlungsanweisungen zur Umsetzung von responsive Webseiten unter Nutzung von verfügbaren Standardmechanismen bei Content Management Systemen sind erstellbar.***
   Das Ziel der Diplomarbeit, allgemeine Handelsanweisungen zu erstellen, kann erreicht werden.

1.3 Motivation und Ziele

Die Präsentation der eigenen Produkte im World Wide Web ist seit einigen Jahren für jede größere Firma von enormer Bedeutung. Um seine Inhalte, Produkte und Präsentationen immer auf dem neusten Stand halten zu können, nutzt man heutzutage CM Systeme zum Verwalten und Managen von Inhalten. Der Anspruch an die Software wächst von Jahr zu Jahr und dies erfordert eine kontinuierliche Softwareanpassung auf die Wünsche der Kunden. Auch im Bezug auf die neuen mobilen Möglichkeiten lässt sich ein Fortschritt der CM Systeme in diese Richtung nicht aufhalten.

Der Bereich Content Management Systeme der T-Systems Multimedia Solutions beschäftigt sich genau mit dieser kundenspezifischen Anpassung von CM Systemen. Ziel der Diplomarbeit soll es sein, eine Bewertung der Standardmechanismen für die Entwicklung von responsive Webseiten zu erstellen. In der Diplomarbeit wird sich auf die drei, im Bereich CMS genutzten, CM Systeme bezogen. Die Bewertung der drei CM Systeme soll abschließend die folgenden zwei Optionen unterstützen. Zum Einen bei der Auswahl des richtigen CM Systems, in Abhängigkeit der Wünsche des Kunden bezogen auf responsive Design. Zum Anderen bei der Bewertung der Kosten für die Anpassung eines bestimmten vom Kunden vordefinierten Systems.

Diese Diplomarbeit richtet sich an alle interessierten Nutzer, die über ein fachliches Verständnis der gewählten Thematik verfügen. Des Weiteren für Frontendentwickler im Bereich Content Management Systeme und für Bereiche, welche die drei gewählten CM Systeme für ihre tägliche Arbeit nutzen.

2 Fachliche Grundlage

2.1 Responsive Design

2.1.1 Bedeutung und Definition

Der klassische Internetnutzer geht auf eine Internetseite um Informationen über ein interessantes Thema zu erhalten. In den meisten Fällen tat er dies bis vor 5 Jahren mit seinen Deskotoprechner oder einen Laptop. Bei diesen Geräten gab es noch keine große Spanne in Bezug auf die Displaygrößen.

Vor circa 5 Jahren begann das Zeitalter der Smartphones und Tablets. Ein Teil der Internetnutzer began das Smartphone oder Tablet für seinen täglichen Internetbesuch zu nutzen. Das Layout der meisten Seiten war für die kleineren Displays nicht mehr optimal. Es stellte sich die Frage, wie kann man denselben Inhalt für mehrere verschiedene Endgeräte möglich machen, wobei dieser größtenteils derselbe bleiben sollte. Diese Eigenschaften lassen sich mit dem englischen Wort „responsive“ gut beschreiben. Übersetzt bedeutet es so viel wie „Auf Jemanden einzugehen“ oder „reaktionsfähig“ zu bleiben. [RD101]

Doch wie sollte man denselben Inhalt verschieden aussehen lassen? Dafür gab es zweierlei Ansätze. Beide nutzten eine Abfrage beim Aufruf einer Unified Resource Locator (URL), welche den Typ des Endgerätes bestimmt. Die Unterscheidung der beiden Varianten lag in der Reaktion auf diese Abfrage. Bei der ersten Variante wurde eine separate Internetseite aufgerufen. Die zweite Variante passte das komplette Layout auf die Displaygröße an und stellte somit die gleiche Seite mit anderem Aussehen dar. In dieser Diplomarbeit behandeln wir die zweite Variante und beschäftigen uns mit der Entwicklung von solchen dynamischen Layouts auf Internetseiten in Abhängigkeit von den Endgeräten. Nehmen wir an, dass ein Internetnutzer eine Internetseite über seinen heimischen Computer besucht. Er kennt die Seite mit einer umfangreichen Navigation auf der linken Seite und einer Hauptseite mit Artikeln, welche in 3 nebeneinander liegenden Spalten zu finden sind. Ein Artikel weckt sein Interesse und er beginnt ihn zu lesen, schafft ihn jedoch nicht vollständig und will ihn später auf dem Weg zur Arbeit zu Ende lesen. Unterwegs öffnet er dieselbe Seite über sein Smartphone. Das Layout unterscheidet sich zu dem vorher aufgerufenen auf dem Computer. Das Smartphone Display bietet wesentlich weniger Platz für die Internetseite. Die umfangreiche Navigation alleine würde schon den ganzen Platz auf dem Display einnehmen und somit keinen Platz mehr für die Artikel auf der rechten Seite bereitstellen.

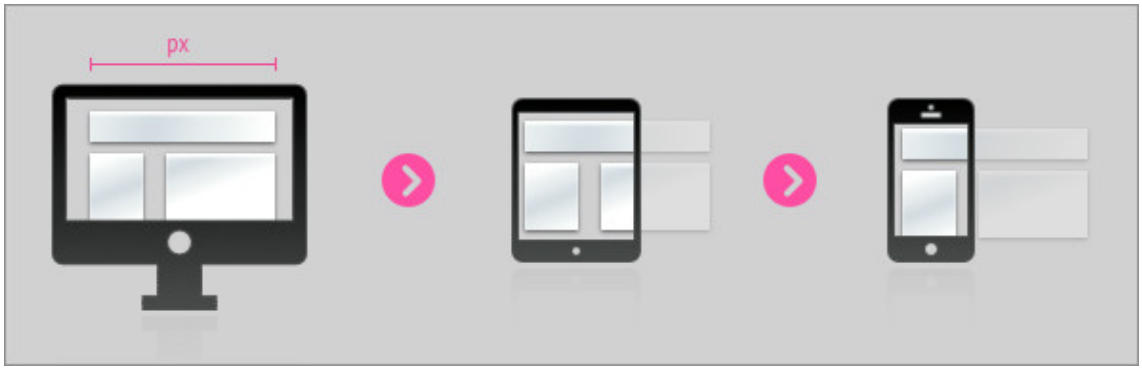

Abbildung 2.1: responsive Design, statische Pixelskalierung, [RD102]

Der Browser erkennt, dass die Internetseite von einem Smartphone aufgerufen wird und verändert das Layout. Die 3 Spalten, welche die Artikel beinhalten, werden nun untereinander dargestellt. Die Navigation könnte zum Beispiel als ausklappbares Menü angezeigt werden, um so das Display nicht mit Informationen zu überladen. Idealerweise findet er den Artikel schnell ohne großen Navigationsaufwand wieder und kann ihn problemlos zu Ende lesen. Ein Artikel mit dem Titel „Responsive Web Design“ von Ethan Marcotte erschien im Mai 2010 für „A List Apart“. Der Gedanke war, eine Website zu entwickeln, welche in mehreren Auflösungen sehr gut darstellbar ist. Er drängte in seinem Artikel Designer dazu, die einzigartigen Charakteristiken des Internets auszunutzen: [RD103]

> *„Das ist unser Weg nach vorne, statt voneinander unabhängige Designs auf eine ständige wachsende Zahl von Geräten zuzuschneiden, können wir sie als unterschiedliche Facetten derselben User Experience behandeln. Wir können für eine optimale Darstellung gestalten, aber gleichzeitig standardbasierte Technologien in unseren Designs einbetten, damit sie nicht nur flexibler werden, sondern sich auch besser an das jeweilige Medium anpassen, auf dem sie dargestellt werden.“* [RD104]

Der Artikel beschreibt anschaulich die Idee von Responsive Webdesign. Hierbei handelt es sich nicht um viele verschiedene Webseiten, die je nach Endgerät aufgerufen werden. Vielmehr ist der Gedanke, eine reine inhaltliche Webseite zu erstellen und das Layout auf die verschiedenen Auflösungen dynamisch anzupassen. Dies bedeutet allgemein gesagt, dass der Browser bei jedem Besuch den Viewport des Endgerätes überprüft und dann sofort das passende Layout über den statischen Inhalt legt. [RD103]

Viewport ist in diesem Fall kein anderes Wort für Browser oder Displaygröße. Vereinfacht gesagt ist der Viewport der Platz, der für die Darstellung eines Webprojekts zur Verfügung steht und genutzt werden kann. [RD102]

Die oben erläuterte Vorgehensweise definiert jedoch nur zum Teil den Begriff responsive Design. Vielmehr handelt es sich bei der Anpassung des Layouts an verschiedene Displaygrößen um ein „adaptives Design“. Der Unterschied zum responsive Design liegt in der Dynamik des Layouts. Beim adaptiven Design werden, wie zuvor beschrieben, verschiedene Sprungpunkte (Breakpoints) gesetzt, bei denen sich das Raster beziehungsweise die Anordnung auf der Seite dem Viewport anpasst. Es wird hierbei jedoch noch die Einheit Pixel für die Größendefinition verwendet. Dies lässt die Seite zwar auf verschiedene Größen reagieren, eine komplett dynamische Anpassung der Spalten geschieht jedoch nicht. [RD102]

Abbildung 2.2: responsive Design, adaptives Design, [RD102]

Erst wenn sich die Größe der Spalten, welche sich je nach Viewport neu positionieren, dynamisch anpasst, handelt es sich um responsive Design. Allgemein beschrieben muss sich die Seite bei der kleinsten Größenveränderung des Viewports an die neue Größe anpassen. Somit wird immer der volle Viewport genutzt und es entstehen keine großen ungenutzten Freiräume.

Abbildung 2.3: responsive Design, responsive Design, [RD102]

2.1.2 Technologie

Die technologische Grundlage des responsive Designs bildet Hypertext Markup Language 5 (HTML5), JavaScript und die dritte Weiterentwicklung von Cascading Style Sheets (CSS3). Die folgenden drei Komponenten sind für die Entwicklung von responsive Designs elementar:

- Media Queries
- Fluid Grids
- skalierbare Bilder

Das Prinzip der Media Queries gab es bereits schon bei CSS2.1, wobei sie damals für eine saubere Print-Darstellung genutzt wurde.

```
<link rel="stylesheet" type="text/css" href="core.css"
  media="screen" />
<link rel="stylesheet" type="text/css" href="print.css"
  media="print" />
```

In dem gezeigten Beispiel wird sowohl eine print.css, als auch eine core.css geladen. Es wird aber nur eine der beiden Dateien interpretiert. Entweder die core.css für die normale Webdarstellung oder die print.css, falls die Webseite auf Papier ausgegeben werden soll.

Mit CSS3 und HTML5 kam eine Reihe von Media Queries hinzu, welche auf eine Vielzahl von Parameter reagieren.

```
<link rel="stylesheet" type="text/css"
  media="screen and (max-device-width: 480px)"
  href="shetland.css" />
```

In diesem Beispiel wird die Shetland.css nur dann interpretiert, wenn der „Screen“ eine maximale horizontale Breite von 480px aufweist, was zum Beispiel bei einem der ersten iPhones der Fall war. [RD104]

Bei der Umsetzung entfernt man sich von der klassischen Pixeldefinition bei Größen und nutzt stattdessen Prozentangaben, um so flexibel auf den Endnutzer und seine Displaygröße eingehen zu können. Bei der Entwicklung von responsive Webseiten sind die nachstehenden drei Säulen wichtig, da sie den Unterschied zur klassischen statischen Entwicklung bilden und dem Frontendentwickler das Anpassen des Layouts wesentlich erleichtern.

Säule Eins - Grids

„Keine festen Layout Grids“ bildet die erste wichtige Säule. Bei der klassischen Entwicklung wurde eine Seite meist in vier Grids aufgeteilt. Diesen wurde anschließend eine feste Pixelbreiten zugewiesen. Bei der responsive Design Entwicklung gibt man den vier Grids einen prozentualen Wert mit. Eine maximale Obergrenze für die gesamte Webseite ist jedoch weiterhin eine notwendige Angabe. Diese sollte sich in der Regel bis maximal 1280px ausbreiten.

Ab einer gewissen Bildschirmbreite lassen sich vier Grids nicht mehr übersichtlich nebeneinander anzeigen. Diese Breite wird mittels der Media Queries gesetzt und eine neue Anordnung der Grids erfolgt. Darüber hinaus werden beispielsweise die ersten beiden Grids nebeneinander mit jeweils 50% Breite angeordnet. Darunter folgen die restlichen beiden Grids mit einer 100%igen Breite nacheinander.

Eine weitere Abstufung der Bildschirmbreite ist erforderlich, da meist in Desktop, Tablet und Smartphone unterschieden wird. Bei dieser werden alle vier Grids untereinander angeordnet und bekommen durchweg den Breitenwert 100%.

Abbildung 2.4: responsive Design, Grids, [RD102]

Das bedeutet, dass sich die Grids zwischen den drei verschieden Begrenzungsbreiten immer in einem festen Raster befinden. Die Breite der Grids passt sich, aufgrund der Prozentangaben, solange der Breite an, bis der nächste sogenannte „Breakpoint“ erreicht ist. An diesem Punkt erfolgt die oben beschriebene Neuanordnung des Rasters. [RD101]

Säule Zwei - Dynamische Schriftgröße

Die zweite Säule behandelt das Thema Schriftgröße und trägt den Titel „keine festen Schriftgrößen“. Diese Säule beinhaltet keine große Dynamik oder Anpassung im Bereich der Anordnung. Sie ist jedoch elementar wichtig für das responsive Webdesign. Ein kleines Beispiel zur Verdeutlichung: Eine Überschrift mit der Größe 14pt lässt sich auf einem normalen Desktop- oder Laptopmonitor gut betrachten. Auf einem Smartphone jedoch gibt diese Schriftgröße nicht mehr den gewünschten Eyecatcher, der auf den folgenden Inhalt aufmerksam machen soll.

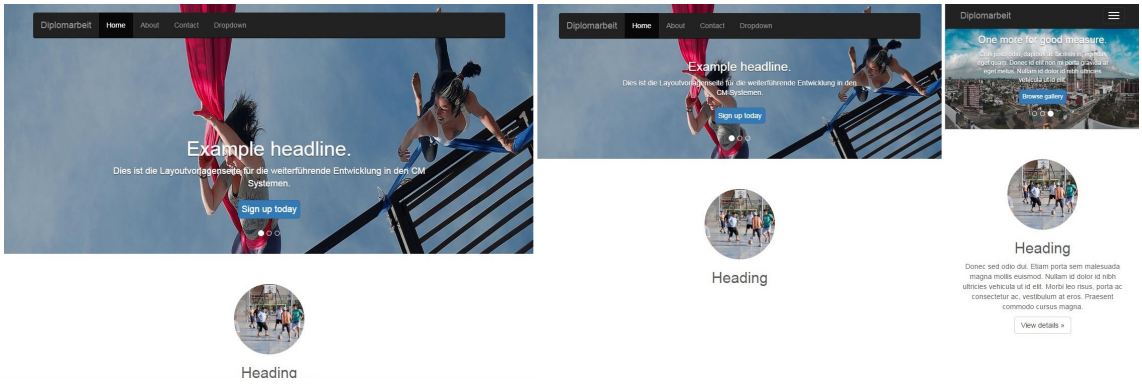

Abbildung 2.5: responsive Design, Überschriften

Es empfiehlt sich hier ebenfalls eine prozentuale Größenangabe. Eine individuelle Anpassung der Schriftgrößen in Abhängigkeit von der Bildschirmbreite ist eine genauso gute Möglichkeit für ein responsive Webdesign, wie auch ein JavaScript Plug-In zum Beispiel FitText, dieses skaliert die Headlines eigenständig.

Abbildung 2.6: responsive Design, Überschriften 2, [RD102]

Säule Drei - Dynamische Bilder

Die letzte Säule beschäftigt sich mit dem optischen Element, den Bildern. Diese Säule ist auch bezüglich der Performance der Seite sehr wichtig und nennt sich „keine festen Bildgrößen“.

Ein tolles Bild einer Landschaft oder eines Produktes wird meist in einem aufwendigen Fotoshooting mit anschließender professioneller Bearbeitung erstellt. Ein so aufwendig erstelltes Bild sollte auf einem hochauflösenden Bildschirm die nötige Größe haben, um das Produkt bestmöglich darzustellen. Auf dem Smartphone kann das jedoch zu extrem langen Wartezeiten führen und das für ein Bild, welches wegen der geringen Bildschirmauflösung unscharf wird.

Performance

Dieses Problem führt uns direkt zu einem weiteren großen Problem. Die Performance ist durch eine Vielzahl von Endgeräten und verschieden aufgebauten Internetanbindungen ein wichtiges Thema bei responsive Webdesign. Es ist weder sinnvoll ein kleines Bild hoch zu skalieren, da es qualitativ nicht mehr ansprechend wirkt, noch in Bezug auf die Performance ein großes Bild zu laden, welches um ein vielfaches verkleinert dargestellt wird.

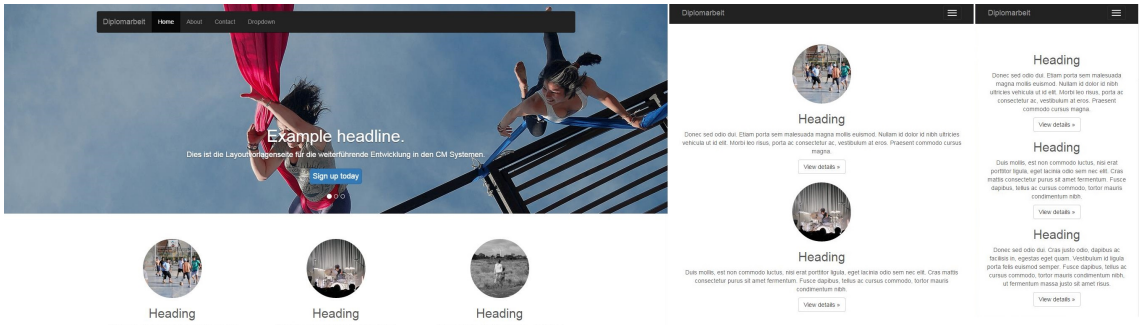

Abbildung 2.7: responsive Design, Bilderskalierung

Mittlerweile gibt es eine Vielzahl von JavaScript Plug-Ins, die abhängig vom Endgerät verschieden große Bilder laden. Das bedeutet, dass auf dem Handy ein kleineres Bild geladen wird um somit nicht zu viel Datenvolumen und Zeit zu verschwenden. Auf dem Desktoprechner kann jedoch das Bild in seiner vollen detailgetreuen hochauflösenden Schönheit betrachtet werden. Es wird mittlerweile eine responsive Fähigkeit in Bezug auf die Internetgeschwindigkeit entwickelt. Bei dieser wird in Abhängigkeit von der Internetgeschwindigkeit verschiedene Bilder geladen.

Eine allgemeine Reduzierung der Webseite im Bezug auf Farben und Hintergrundbilder ist ebenfalls sinnvoll, um eine bessere Performance bei mobilen Layouts zu gewährleisten. Mit den Media Queries ist es auch möglich verschiedene JavaScript Dateien zu laden, um auch da eine abgespeckte Variante für mobile Geräte zu erstellen.

Barrierefreiheit

Das Thema Barrierefreiheit spielt im Internet eine immer größere Rolle und wird bei der Erstellung von Webseiten immer wichtiger. Barrierefreiheit bedeutet alles für jedermann erreichbar zu machen.

Die Internetseite „barrierefrei informieren und kommunizieren“ bietet ein Prüfverfahren zum Testen der Barrierefreiheit. Es wurden die einzelnen Punkte dieses Verfahrens untersucht und eine Referenz zum responsive Webdesign gezogen.

Das Prüfverfahren besteht aus einer Übersicht von relevanten Punkten, welche für eine barrierefreie Internetseite wichtig sind. Die Übersicht ist im Anhang und unter der Quelle [RD106] zu finden.

Bei der Erstellung von responsive Design werden die folgenden Punkte der Übersicht bereits unterstützt.

1.3.1g Kein Strukturmarkup für Layouttabellen
Bei responsive Design wird auf die Verwendung von Tabellen zum Erstellen des Layouts komplett verzichtet. Die Positionierung und Strukturierung wird ausschließlich mit Grids vollzogen

1.4.4a Schriftgröße variabel
Dynamische Schriftgrößen ist eines der oben beschriebenen Säulen bei responsive Design. Es erfolgt stets eine Anpassung der Schriftgrößen auf den entsprechenden Viewport.

1.4.4b Bei Zoom auf 200% benutzbar
Beim Zoomen auf Webseiten, wird der Viewport des Browsers verändert und somit passt sich die responsive Webseite automatisch auf den neuen Viewport an.

2.1.1a Ohne Maus nutzbar
Responsive Design schließt eine Anpassung für Smartphones und Tablets mit ein. Diese Geräte werden meist ohne eine Maus gesteuert.

2.1.3 Entwicklungsverlauf ohne CMS

Die Planung ist das wohl wichtigste Element bei der Entwicklung von responsive Webseiten. Am Anfang muss zunächst das Konzept stehen, wie die Seite in den verschiedenen Darstellungsmethoden aussehen soll.

Die „Mobile first“ Maxime empfiehlt sich bei der Entwicklung von responsive Webseiten. Die kleinste Darstellungsform sollte als Erstes geplant und umgesetzt werden.

Anschließend arbeitet man sich mittels des sogenannten Bottom-Up Vorgehens in der Auflösung immer weiter nach oben. Der Grund für diese Herangehensweise liegt in folgender einfacher Feststellung: Es ist deutlich einfacher, Details hinzuzufügen als diese später entfernen zu müssen. [RD105]

Der klassische Workflow bei der Entwicklung von Webseiten besteht aus Konzept, Umsetzung und Design. Auf dieses wird bei responsive Wegdesign verzichtet und auf neue und angepasste Workflows zurückgegriffen.

Wie schon beschrieben, bildet die Planung das Kernstück und steht am Anfang der Entwicklungsreihe. Bei allen Layouts im World Wide Web geht es darum, den Inhalt möglichst perfekt in Szene zu setzen und ihn somit schnell dem Endnutzer zu präsentieren. Daher sollte während der Planungsphase klar definiert werden, worum es auf der Webseite gehen soll und wo die wichtigen Punkte auf dieser angeordnet werden soll.

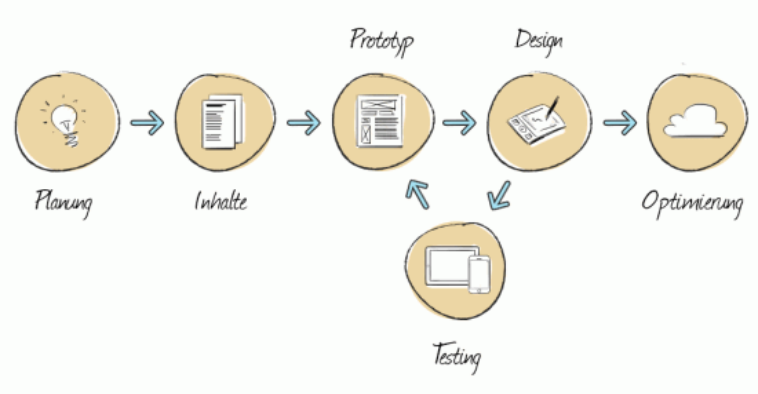

Abbildung 2.8: responsive Design, Workflow, [RD105]

Nach der Konzeption beginnt die Umsetzungsphase. Ab diesem Punkt weicht der Workflow von der klassischen Variante ab. Es wird in der Folge eine Prototypseite entwickelt, auf welcher das Verhalten der einzelnen Komponente definiert wird. In dem Prototyp sollten sich die Komponenten schon je nach Viewport verschieben können und auch die Größe der einzelnen Elemente sollte sich dynamisch anpassen. Nach Fertigstellung des Prototyps lässt sich das allgemeine Verhalten der Seite erkennen. Das Design wird anschließend über die einzelnen Komponenten gezogen. Damit sich das Layout dynamisch verhält, muss die Webseite aus vielen einzelnen Komponenten bestehen. Diese sollten alleinstehend und abgeschlossen entwickelt werden, um sie somit als Ganzes auf der Seite verschieben zu können. Mit Hilfe des Prototypen weiß der Designer nun genau, welche Elemente flexibel gestaltet werden müssen und welche nicht. Fehlerhaftes Verhalten des Prototyps oder des Designs werden sofort erkannt und

können dementsprechend angepasst werden.

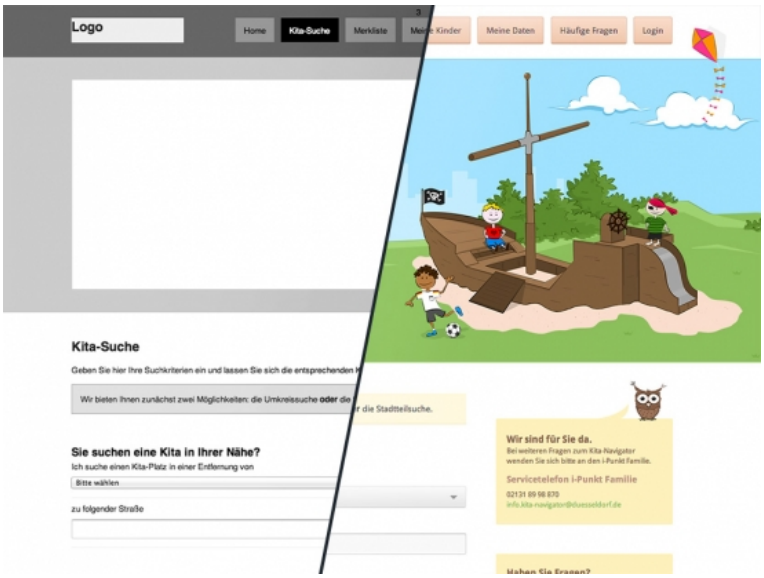

Abbildung 2.9: responsive Design, Prototyp und Design im Vergleich, [RD105]

Diese drei Teile des Workflows können bei der Entwicklung mehrmals durchlaufen werden. Idealerweise arbeitet der Entwickler des Prototyps und der Designer eng zusammen oder sind am besten ein und dieselbe Person. Abschließend sind nur noch kleinere Optimierungen von Nöten, sollten besondere Inhalte auf der Seite neu implementiert werden.

Ein gut durchdachtes und geplantes responsive Webdesign sollte keine größere Wartung mehr beanspruchen, da sich die Seiten bei allen Eventualitäten eigenständig anpassen.

2.2 Content Managemant System

2.2.1 Bedeutung und Definition

In diesem Abschnitt soll dem Leser dieser Arbeit ein kurzer Überblick über den allgemeinen Begriff Content Management System gegeben und weiterführend eine kurze Vorstellung der drei ausgewählten Content Management Systeme vorgenommen werden. Diese drei Systeme, welche in der Arbeit untersucht werden, sind die, im Bereich Content Management in der T-Systems Multimedia Solutions GmbH, am häufigsten genutzten Content Management Systeme. Es wird bei der späteren Betrachtung dieser Systeme das Hauptaugenmerk auf die Features für die Layoutumsetzung gelegt und darauf, wie geeignet sie bei der Entwicklung von responsive Webdesign sind.

Ein Content Management System ist eine Software zur Erstellung, Verwaltung und Präsentation von redaktionellen Inhalten. Dabei sind die Hauptfunktionen eines CMS das organisierte gemeinschaftliche Sammeln, Strukturieren und Aufbereiten für verschiedene Ausgabekanäle, sowie das Ausliefern von Informationen mit verschiedenen vordefinierten Workflows. Ziel dieser Systeme ist es, technisch weniger versierten Benutzern zu ermöglichen, ihre Inhalte zu erstellen, bearbeiten, organisieren und ins World Wide Web oder beliebige Ausspielkanäle zu bringen. Große Mengen von Inhalten können auf diese Weise gemanagt werden. Dies erfordert bei der Verwendung von CMS keine Kenntnisse über Webprogrammierung oder Sprachen wie HTML. Die Inhalte, beispielsweise in Form von Text- oder Multimedia-Dokumenten, werden im weiteren Verlauf als „Content“ bezeichnet.

Der Begriff CMS kann nicht eindeutig definiert werden. Da es bei der Entwicklung von CMS zunächst keinen einheitlichen Ausgangspunkt gab, existierten keine Spezifikationen oder Beschreibungen, nach welcher die Systeme entwickelt werden konnten. Verschiedene Hersteller arbeiteten parallel an ihren Lösungen, wobei der Schwerpunkt aufgrund unterschiedlicher Anforderungen an verschiedenen Stellen lag. Aus diesem Grund ist eine klare Abgrenzung verschiedener CMS-Typen nur sehr schwer möglich. Das Feld der Content Management Systeme lässt sich jedoch in die folgenden drei Systeme aufteilen. Web Content Management System (WCMS) dienen ausschließlich zum Darstellen von Content im Internet und werden bei der Anwendung in Medienbereichen auch als „Redaktionssysteme“ bezeichnet. Kommerzielle Lösungen für große Unternehmen mit einer umfangreichen Menge an Content werden Enterprise Content Management System (ECMS) genannt. Außerdem existieren auch CMS zur Verwaltung und gemeinsamen Nutzen von Dokumenten – sogenannte Document Management System (DMS).

Die wichtigsten Punkte bei CM Systemen liegen in der Unabhängigkeit der Ausspielkanäle, die Trennung von Design und Content, sowie die „medienneutrale Datenhaltung“. Der Begriff „medienneutral“ wird bei Daten genutzt, welche nicht für eine spezielle Ausgangsbedingung vorbereitet sind, sondern als Grundlage für viele verschiedene Ausgangssituationen dienen können. [CMS01]

Bei Content Management Systemen kann der vom Autor erstellte Inhalt auf zwei verschiedene Weisen dargestellt werden. Zum Einen als HTML-Seite im Internet und zum Anderen als Portable Document Format (PDF), Extensible Markup Language (XML) oder ähnliches. Das endgültige Format wird zum Teil erst durch die entsprechende Abfrage an das System generiert.

Ein weiterer wichtiger Punkt bei CM Systemen beinhaltet die Rechteverwaltung. Je nach seiner Rolle kann ein Benutzer Inhalte erstellen, bearbeiten oder löschen sowie wichtige Systemeinstellungen anpassen, um so eine breite Verteilung der Arbeitsaufgaben abhängig von den Kenntnissen der Benutzer zu ermöglichen. Mit einem solchen

Rechtesystem kann man außerdem verschiedene inhaltliche Bereiche für verschiedene Rechtegruppen sichtbar oder unsichtbar machen. Bei ECM Systemen wird außerdem den Nutzern die Möglichkeit der Versionierung von Inhalten an die Hand gegeben. Somit können Änderungen zu jeder Zeit rückgängig gemacht werden oder der Inhalt auf einen älteren Zustand zurückgesetzten werden.

2.2.2 Allgemeine Vorstellung der Content Management Systeme

Adobe Experience Manager (AEM) ist eines der drei Systeme, die im Zuge dieser Arbeit analysiert und angepasst wird. AEM ist ein umfangreiches CMS, welches der Nachfolger des erfolgreichen Adobe Communique (Adobe CQ) ist und die sechste Version der Software darstellt. Ursprünglich wurde CQ durch die Schweizer Firma Day entwickelt und von Adobe 2010 aufgekauft. Aus diesem Grund ist es weitläufig auch noch unter dem Namen Adobe Day bekannt. [CMS02]

Das Produkt AEM richtet sich in erster Linie an große Unternehmen und Konzerne mit umfangreicher und oft globaler Infrastruktur. Es vereint Web Content Management, Digital Asset Management und Social Collaboration. Auf diese Weise ermöglicht das CMS den Kunden die Verwaltung großer Informationsmengen, mehrerer interner und externer Webseiten, großer Mengen von Multimedia-Daten und detailliertes Workflow-Management. Zu den Kunden in Deutschland zählen unter anderen Boehringer Ingelheim und die Carl Zeiss AG.

Das zweite Content Management System, das in dieser Arbeit betrachtet wird, ist CoreMedia Blueprint. Die Firma CoreMedia wurde 1996 in Hamburg von vier Mitarbeitern der Technischen Universität Hamburg gegründet. CoreMedia zählt zu den ECMS, da es eher für Großkunden geeignet ist. Besonderen Wert wird hierbei auf Hochverfügbarkeit, Verarbeitung von Seiten-Aufrufzahlen in Millionenhöhe am Tag und Nutzung durch viele Redakteure gelegt.

Zu den ersten Kunden zählte die Deutsche Presse Agentur und der Axel Springer Verlag. 2000 folgte die Deutsche Telekom mit ihrem T-Online Portal. Im Jahr 2002 wurde im Zuge der „Bund 2005 Initiative“ CoreMedia als Standard CM System für alle deutschen öffentlichen Behörden ausgewählt. Im Februar 2013 veröffentlichte CoreMedia ihre neuste Version der Software namens CoreMedia Blueprint. Dies ist die siebte Version des Produktes. [CMS03]

Das letzte CMS ist Magnolia 5. Das Produkt Magnolia wird seit 2003 entwickelt und ist als reine Open-Source-Software gestartet. 2006 wurde zu der Open-Source Community-Edition noch eine weitere kostenpflichtige Enterprise Version verfügbar, welche erweiterte Funktionen und Support im Vergleich zu der Open-Source-Software bietet.

Die aktuelle Version wurde im Juni 2013 unter den Namen Magnolia 5 veröffentlicht. REWE, Red Bull und Sony sind nur einige der Unternehmen, welche Magnolia als CM System nutzen, um die Umsetzung ihres Webauftritts damit zu realisieren. [CMS04]

Alle drei, als ECMS geltenden, vorgestellten Content Management Systeme basieren auf Java und sind für größere Unternehmen geeignet. Alle Produkte bieten eine softwareinterne Möglichkeit das Layout mittels vordefinierter Templates und durch das Hinzufügen von benutzerdefinierten CSS zu editieren. Das Anpassen der Templates ist ebenfalls bei allen Systemen möglich, da den Entwicklern ein voller Codezugriff ermöglicht wird.

Alle drei Produkte haben sich stark in ihrer neusten Version mit dem Thema responsive Webdesign beschäftigt und verschiedene Möglichkeiten der Umsetzung implementiert. Im dritten Kapitel werden die Produkte speziell auf diese Features und die Umsetzung von responsive Webdesign untersucht.

3 Untersuchung der responsive Fähigkeiten

3.1 Methoden

Zur Erstellung dieser Arbeit wurden die nachfolgenden wissenschaftlichen Methoden verwendet:

3.1.1 Recherche

Eine gezielte Suche nach Informationen zu einem festgelegten Thema bezeichnet man als Recherche. Im wissenschaftlichen Kontext wird Recherche folgendermaßen aufgefasst:

- Nachforschungen
- Beschaffung von Informationen
- Systematische Erschließung
- Kennenlernen von Hintergründen und Umständen
- Sich eine eigene Sichtweise bilden

Der Prozess des Recherchierens umfasst folgende Phasen:

1. Vor der Recherche:
    - Erstellung einer detaillierten Problembeschreibung
    - Gegebenenfalls Formulierung einer Recherchethese
    - Gliederung, Ablaufplan
2. Während der Recherche:
    - Permanentes Integrieren neuer Informationen
    - Überprüfung neuer Informationen, Überarbeitung und gegebenenfalls Veränderungen oder Neufassungen vornehmen
3. Nach der Recherche:
    - Informationsauswertung
    - Zielfixierung und endgültige Problembestimmung/-gewichtung
    - Aufstellen des endgültigen Ablaufplanes [ME01]

Die durchgeführte Recherche umfasst sowohl eine Literatur- als auch eine Internetrecherche. Es wurde bei der Literaturrecherche lediglich auf E-Books zurückgegriffen. Die Durchsuchung des World Wide Web konzentrierte sich in erster Linie auf Internetauftritte, Wikis oder Whitepapers. Die genutzten Quellen sind im Literaturverzeichnis aufgeführt und die entsprechenden Passagen im Text mithilfe von eckigen Klammern und einem eindeutigen Kürzel für jede Quelle gekennzeichnet, wie beispielsweise die oben genannte [ME01]. Die Methode wird in erster Linie im Grundlagen-Teil sowie bei allen theoretischen Betrachtungen verwendet.

3.1.2 Modellierung

Modelle dienen in der Wissenschaft der vereinfachten Darstellung komplexer Strukturen und Vorgänge. Speziell in der Informatik bilden sie einen Realitätsausschnitt, um auf diese Weise eine Aufgabe mithilfe von Informationsverarbeitung zu lösen. Derartige Modelle werden auch als Domänenmodelle bezeichnet. Typische Beispiele für Domänenmodelle sind Modelle, welche die zu erstellende Software sowohl architektonisch (Architekturmodell) abbilden, als auch deren Programmcode (zum Beispiel Programmablaufpläne) oder ihre Datenhaltung beschreiben. [ME02]

Als Modell dient bei dieser Diplomarbeit ein Bootstrap Layout von der Seite getbootstrap.com. Dieses Modell beinhaltet die klassischen Features von modernen Produkt- und Präsentationsseiten. Die Beispielseite wurde noch mit Texten und nicht geschützten Bildern optisch aufgewertet.

3.1.3 Prototypische Implementierung

Prototyping ist eine Methode der Softwareentwicklung, bei der ein lauffähiger Auszug oder eine anderweitige Modellierung einer Softwarekomponente implementiert wird, um schnell erste Ergebnisse zu erhalten und die Eignung dieses Lösungsansatzes überprüfen zu können. Dies erfordert wesentlich weniger Aufwand und Zeit als die Erstellung der voll lauffähigen Komponente. Es gibt dabei mehrere Typen:

- Exploratives Prototyping: zur Bestimmung von Anforderungen und zur Beurteilung bestimmter Problemlösungen, Konzentration auf Funktionalität
- Evolutionäres Prototyping: Entwicklung der Grundfunktionalitäten zur Überprüfung der Akzeptanz beim Nutzer und der Notwendigkeit von ergänzenden Funktionen, Weiterentwicklung des lauffähigen Prototyps zur Produktreife
- Experimentelles Prototyping: experimenteller Prototyp zum Sammeln von Erfahrungen mithilfe von Problemanalysen und Systemspezifikation, die Ergebnisse werden zur Entwicklung des Produktes verwendet
- Rapid Control Prototyping: Softwareentwicklung von Regelungen und Steuerungen mit Hilfe grafischer Tools

- Vertikales Prototyping (Durchstich): Implementierung eines ausgewählten Systemteils durch alle Ebenen hindurch zur Anfertigung eines konkreten Programmteils, geeignet bei ungeklärten Funktionalitäts- oder Implementierungsfragen in anderen Programmteilen
- Horizontales Prototyping: Fertigstellung einer ausgewählten Ebene des Gesamtsystems zur Präsentation oder Orientierung für andere Ebenen (Beispiel: Dummylink oder ähnliches)

Bei der prototypischen Implementierung handelt es sich im Zuge dieser Diplomarbeit um eine Implementierung einer responsivefähigen Prototypseite in drei verschiedene CM Systeme. Hierbei wird versucht so viele Standardmechanismen wie möglich zu nutzen und lediglich eine Anpassung vorzunehmen, welche im Zuge einer Frontendentwicklung standardmäßig durchgeführt werden kann.

3.2 Magnolia 5

Magnolia 5 unterscheidet sich im ersten Eindruck schon von den beiden anderen CM Systemen. Der Startbildschirm ist sehr übersichtlich und mit einem Kachelsystem sehr strukturiert aufgebaut und teilt sich in drei Bereiche auf. Der Erste ist der Edit-Bereich, in dem alle inhaltlich relevanten Einstellungen vorgenommen werden. Dieser Bereich wird vorwiegend von den Redakteuren genutzt. Im zweiten Abschnitt finden Administratoren alle nötigen Einstellungsoptionen und können das CM System nach belieben konfigurieren.

Der letzte Abschnitt beinhaltet entwicklerseitige Themen. Hier lassen sich Templates bearbeiten, CSS- und JavaScript-Dateien erstellen. Des Weiteren gibt es unter diesem Punkt noch weitere Möglichkeiten die Seiten kundenspezifisch anzupassen. Das Einsehen von Log-Dateien sowie die Rechteverwaltung sind weitere Möglichkeiten, welche sich im dritten Abschnitt der Startseite befinden.

Magnolia bietet für die Erstellung von Layouts und Seitendesigns die Möglichkeit diese mit Hilfe von Themes zu realisieren. In der Community Version des CM Systems ist eine Multimandantenverwaltung nicht möglich, da bei dieser Version nur ein Theme konfiguriert und benutzt werden kann. Das bedeutet es ist lediglich möglich, nur einen vollständigen Webauftritt komplett nach seinen Vorstellung anzupassen. Möchte man noch einen weiteren Webauftritt hinzufügen und diesen anders darstellen, sollte man auf die Enterprise Version zurückgreifen.

Das Thema Themes ist sogleich auch das erste hervorzuhebende Feature, welches den Nutzer bei der Realisierung von responsive Webseiten unterstützt. Die Themes-Ordner beinhalten vier Unterordner. In den ersten beiden Ordnern wird definiert, welche CSS- und JavaScript-Dateien für das Theme genutzt werden sollen. Bei den letzten zwei

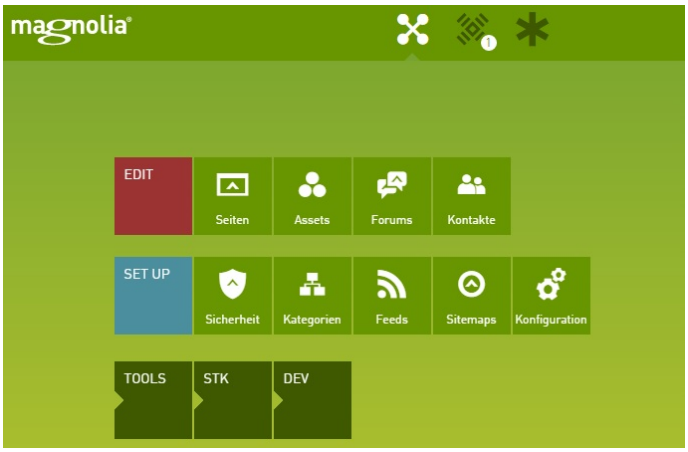

Abbildung 3.1: Magnolia 5, Startbildschirm

Unterordnern, handelt es sich um den „Imaging“ und „bodyClassResolver“ Ordner. Letzterer beinhaltet einen Verweis auf die genutzte Javaklasse, welche zum Rendern der Seite verwendet werden soll. Der „Imaging“ Ordner, bietet den Entwicklern eine komfortable Möglichkeit die Performance seiner Seite zu verbessern und ist ein hilfreiches Feature für responsive Design, welches in der Folge noch näher erläutert wird.

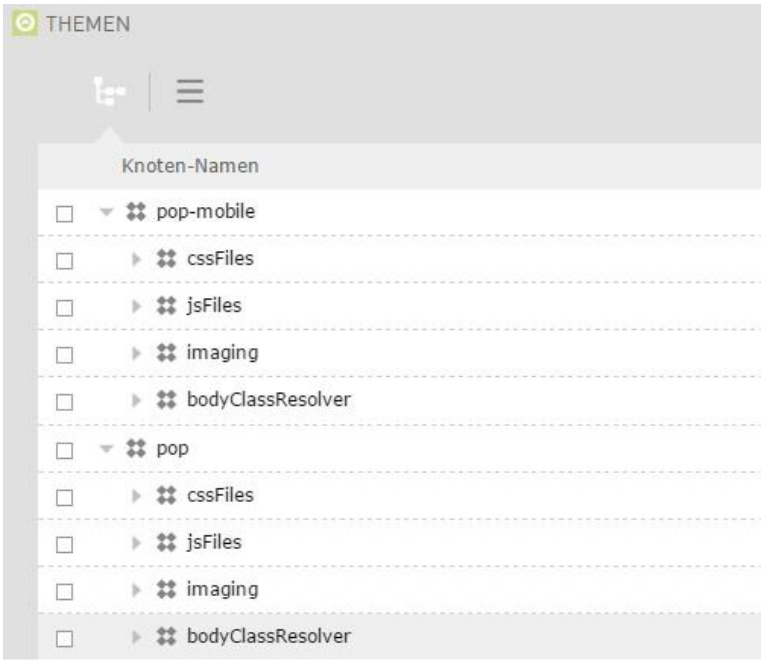

Abbildung 3.2: Magnolia 5, Themen

Die eigentliche Besonderheit an dem Theme-Model befindet sich unter dem Menüpunkt „Seitendefinition“. Es ist möglich für die Seite verschiedene Variationen einzustellen. Das bedeutet, die Seite lädt sich in Abhängigkeit von dem ermittelten Endgerät mit

verschiedenen Themes und somit auch mit unterschiedlichem Aussehen. Bei Magnolia sind zwei Variationen auswählbar. Zum einen die Smartphone und zum anderen die Tablet Variante. Hierbei sind softwareseitig vordefinierte Viewports festgelegt, bei denen die verschiedenen Variationen genutzt werden. Sollte keine Variation definiert worden sein, werden immer die Standardeinstellung der Seite für alle Viewports genutzt. Unter den Variationen lassen sich nicht nur Themes definieren, man kann auch ganze Bereiche der Seite ausblenden. Dies geschieht nicht mit Hilfe von CSS-Befehlen, sondern bereits beim Rendern der Seite. Der ausgewählte Bereich wird nicht mit ausgeliefert.

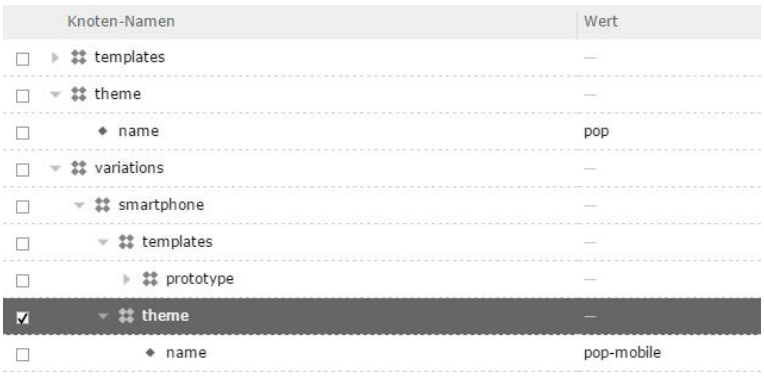

Abbildung 3.3: Magnolia 5, Seitendefinition

Magnolia bietet den Frontendentwicklern noch weitere Unterstützungen im Bezug auf responsive Design an. Es ist möglich bei der Einbindung von CSS-Dateien, diese direkt mit Media Queries zu verbinden. Das bedeutet, dass bestimmte CSS-Dateien nur bei bestimmten Viewports geladen und genutzt werden. Das bietet dem Entwickler die Möglichkeit das Thema responsive Design sehr übersichtlich und leicht verwaltbar anzulegen.

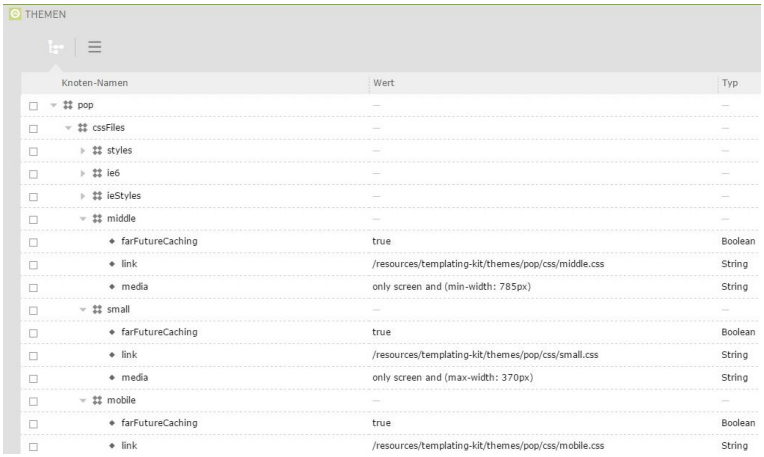

Abbildung 3.4: Magnolia 5, CSS Einbindung

Magnolia bringt bereits mit den oben genannten Features ein Fülle von guter und

einfacher Unterstützung bei der Entwicklung von responsive Webseiten mit. Die letzten beiden Features, mit denen Magnolia überzeugen kann, beinhalten die Performance und Vorschaufunktion.

Die Vorschaufunktion ist ein gängiges Mittel in CM Systemen, da hier dem Redakteur die Möglichkeit geboten wird, seine redaktionierten Inhalte unter simulierten Livebedingungen zu sehen. Bisher war dies ein einfaches Umschalten zwischen dem Edit und Previewmodus, welches die Seite ohne die einzelnen Editiermöglichkeiten darstellte. Magnolia geht hierbei noch einen Schritt weiter. Mit der Version 5 des CM Systems ist es möglich zwischen mehreren Geräten umschalten zu können.

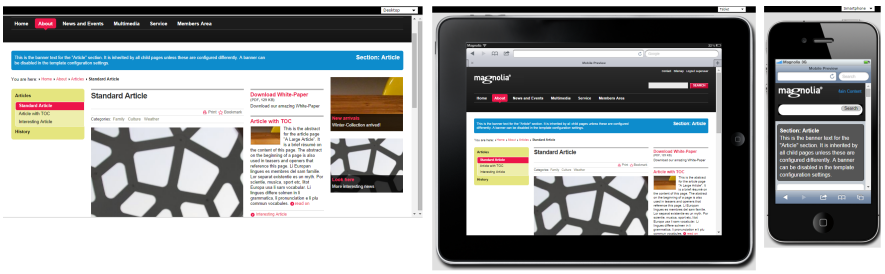

Abbildung 3.5: Magnolia 5, Previewmodus

Es ist möglich direkt beim Editieren der Seite, diese sich auf einen simulierten Smartphone oder Tablet anzeigen zu lassen. Einzige Einschränkung bei dieser Funktion ist die vordefinierte und nicht editierbare Auswahl der Geräte. Es ist nicht möglich spezielle Geräte mit variierendem Viewport einzubinden.

Das letzte und innovativste Feature, das Magnolia den Kunden an die Hand gibt, befasst sich mit dem Thema „Imaging“. Wie im ersten Abschnitt des Kapitels bereits erwähnt ist dieses unter den einzelnen Themen zu verwalten. Das Prinzip der „Imaging“-Funktion ist es, Bilder, die für einen kleinen Viewport genutzt werden, nicht in ihrer vollen Größe auszuliefern.

Ein Bild mit einer Originalgröße von 1900x1900 Pixel ist in diesem Fall 1.2 Megabyte groß. Solch ein Bild auf einem Handy laden zu lassen, um es anschließend mit 300x300 Pixel darzustellen, bringt performancemäßige Probleme mit sich und mindert die Freude beim Aufruf der Seite auf mobilen Geräten. Das „Imaging“-Feature von Magnolia, greift genau an diesem Punkt an. Sollte der Frontendentwickler das gesamte Themenkonzept nutzen, kann er mit Hilfe dieses Features die Bilder gerätespezifisch in verschiedenen Größen publizieren. Das Bild würde nach der Aktivierung auf einem Smartphone zum Beispiel nur noch in der Größe 300x300 Pixel und auf einen Tablet mit 700x700 Pixel geladen werden und somit eine Dateneinsparung auf dem Smartphone von 0.9 Megabyte (75%) und auf dem Tablet von 0.6 Megabyte (50%) nach sich ziehen.

Für Webseiten mit einer Vielzahl von Bildern ist dieses Feature ideal geeignet, um auch

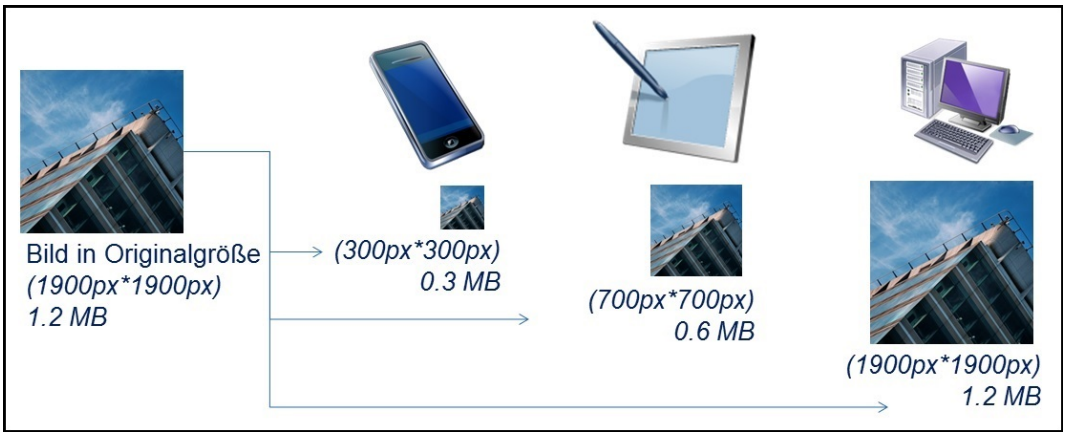

Abbildung 3.6: Magnolia 5, Imaging

auf mobilen Geräten den Besuch dieser Seite genießen zu können und nicht mit langen Wartezeiten beschäftigt zu sein.

Magnolia 5 liefert standardmäßig einen vordefinierten und bereits mit Beispielinhalt gefüllten Webauftritt, welcher sich responsive verhält. Dieser nutzt bereits alle bisher aufgeführten Features und verwendet die ebenfalls vordefinierten Themes „Pop“ und „Pop-mobil“.

3.3 CoreMedia Blueprint

CoreMedia 7 Blueprint nutzt im Vergleich zu den anderen beiden CM Systemen ein anderes Redaktionierungssystem. Blueprint und seine Vorgänger legen ihr Hauptaugenmerk auf die Referenzierung und Nutzung von Artikeln und Teasern. Das bedeutet, es werden immer nur Elemente von Seiten erstellt und zentral abgelegt. Diese können dann auf einer Vielzahl von Seiten als unterschiedlicher Typ eingebunden werden.

Der Inhalt passt sich je nach Typ an das vordefinierte Layout des ausgewählten Types an und wird auf der Seite gerendert. Bei den anderen beiden CM Systemen werden die Seiten meist der Reihe nach separat aufgebaut. Lediglich Elemente wie ein Kontaktformular oder welche, die öfter auf verschiedenen Seiten Verwendung finden, werden auch zentral redaktioniert und referenziert.

Auf der CoreMedia Startseite wird nicht wie bei Magnolia zwischen verschiedenen Bereichen unterschieden. Es gibt eine Bibliothek in der alle Dateien zu finden sind. In der linken Menüleiste sind Verknüpfungen zu bestimmten Ordnern angegeben, welche häufiger verwendet werden.

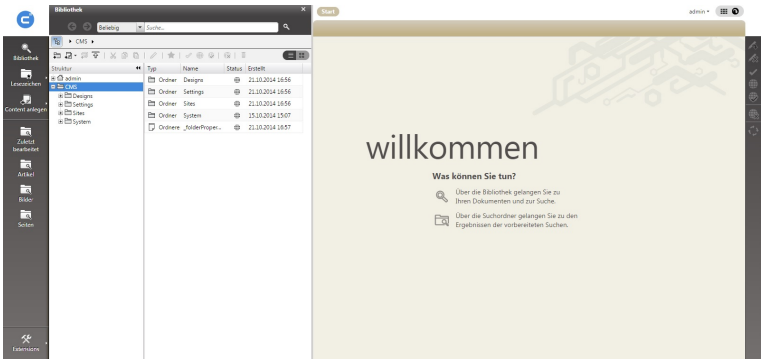

Abbildung 3.7: CoreMedia 7 Blueprint, Startseite

In Sachen Features bietet CoreMedia bei Weitem nicht eine solch gute Unterstützung für die Entwicklung von responsive Design wie Magnolia. Herausstechende Features wie das „Imaging“ oder die Einbindung von CSS Dateien in Abhängigkeit von Media Queries sucht man bei CoreMedia genau so , wie die dynamische Templateanpassung je nach Endgerät.

Alle Einstellungen werden in der Startseite des jeweiligen Mandanten getroffen. Die CSS- und JavaScript-Einbindungen sind ebenso unrevolutionär, wie auch der Previewmodus von CoreMedia. Diese einfache und konservative Methode der Einbindung hat aber den positiven Nebeneffekt, dass es auf allen Unterseiten ebenso möglich ist neue CSS- und JavaScript-Dateien hinzuzufügen. Man kann somit seitenspezifische Layouts erstellen, ohne dabei die CSS Klassen in den Templates zu verändern. Dies ist bei Magnolia und AEM 6 nicht so einfach möglich und benötigt einen größeren Aufwand seitens der Frontendentwickler.

Abbildung 3.8: CoreMedia 7 Blueprint, Einstellung

CoreMedia bietet zwar in Sachen responsive Design Entwicklung im üblichen Sinne keine großen Features, die einen dabei unterstützen, jedoch gibt es mit der sogenannten Personalisierung dem Begriff responsive Design eine ganz andere Richtung. Dieses Feature macht es möglich, die Start- und Unterseiten von Webauftritten dynamisch, sozusagen responsive, an den Besucher der Seite anzupassen.

Das Prinzip funktioniert wie folgt. Ein Nutzer, welcher eine Seite häufiger besucht und auf dieser angemeldet ist, bekommt anhand von Cookies gewisse Interessentags. Jeder Artikel in CoreMedia kann mit Tags versehen werden. Anhand dieser Artikel und Nutzertags wird nun eine Seite dynamisch für den Nutzer in Abhängigkeit seiner Interessen erstellt. Dies ist natürlich ein optionales Feature. Es ist immer noch möglich statische Seiten zu redaktionieren, ohne das sich der Inhalt automatisch dem Nutzer anpasst.

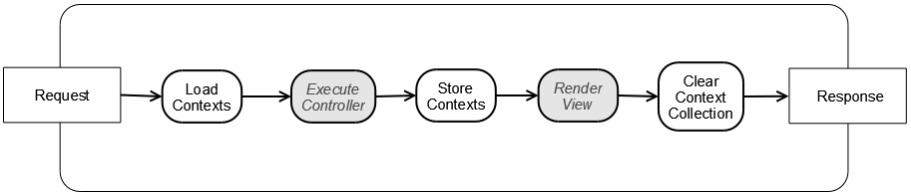

Abbildung 3.9: CoreMedia 7 Blueprint, Personalisierung

Aufgrund des abweichenden Radaktionsprinzips bei CoreMedia zeigt sich, anders als bei den beiden Vergleichssystemen, jedoch immer sofort jede Änderung in der simulierten Liveansicht. Diese befindet sich rechts neben den Seiteneinstellungen und aktualisiert sich bei jeder Komponentenänderung oder Anpassung automatisch. Ein Umschalten zwischen verschiedenen Endgeräten ist bei CoreMedia jedoch nicht implementiert. Lediglich ein Preview-Link mit dazugehörigem Sicherheitscode lässt sich nutzen, um die Seiten auf mobilen Endgeräten sichtbar zu machen, ohne dass diese schon im World Wide Web für jeden sichtbar sind.

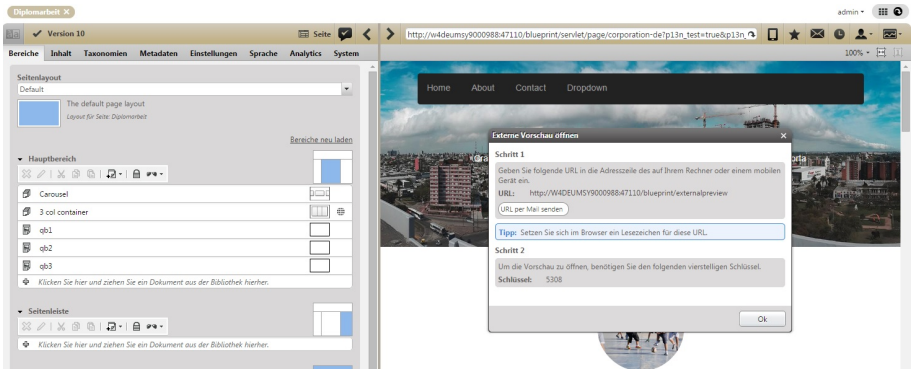

Abbildung 3.10: CoreMedia 7 Blueprint, Previewansicht

Wie oben bereits beschrieben, bietet CoreMedia 7 Blueprint in Sachen „Imaging“, nicht wie Magnolia 5, Möglichkeiten die Performance der zu redaktionierenden Seite zu optimieren. Ganz kann man CoreMedia diese Unterstützung nicht absprechen, da es eine standardmäßig implementierte Funktion gibt. Jedes Bild, welches man in CoreMedia hochlädt, wird sofort auf vordefinierte Seitenverhältnisse mit dazugehöriger Größe angepasst. In Abhängigkeit vom Typ des Objektes, welches das Bild verwenden wird (Artikel, Teaser et cetera) nutzt das System verschiedene Bildausschnitte. Diese können mittels eines Auswahlrechteckes definiert werden.

Jedes Seitenverhältnis hat im System eine zugehörige maximale Auflösung, was verhindern soll, dass zu große Bilder in einer kleinen Ansicht genutzt werden. Lädt man ein Bild mit einer Auflösung 1280x1270 Pixel hoch, skaliert es CoreMedia automatisch, dem Seitenverhältnis angepasst, herunter. Somit kann man von einem repsonsiven Verhalten im Bezug auf die Objekte, welche die Bilder verwenden, sprechen.

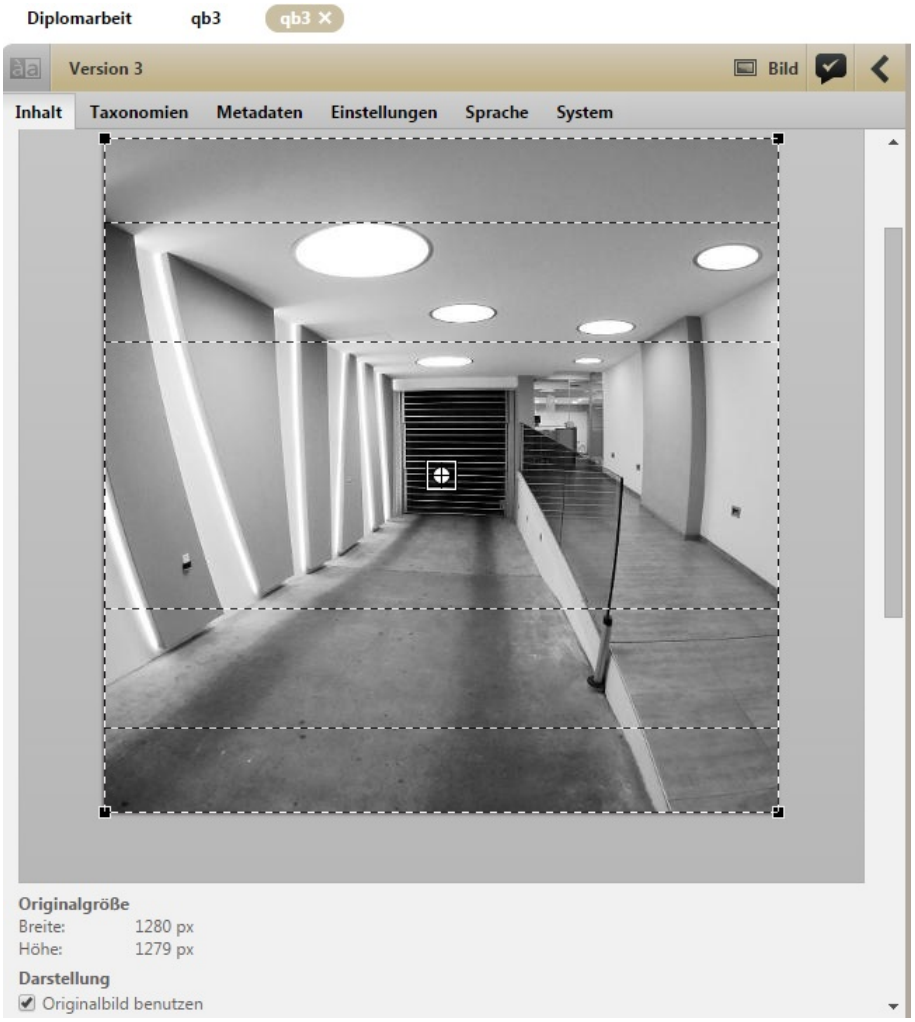

Abbildung 3.11: CoreMedia 7 Blueprint, Imaging

Sollte man die Bilder in der vollen Auflösung nutzen wollen, ist es möglich

dieses mittels einer Checkbox auszuwählen. Die vordefinierten Auflösungen lassen sich leider nicht mandantenspezifisch anpassen. Die Nutzung dieser Funktion zur Performanceoptimierung macht demnach nur Sinn, wenn man die Bilder nicht über eine bestimmte Auflösung hinaus benutzen will.

3.4 Adobe Experience Manager 6

Das in der Analyse als letztes überprüftes CM System AEM 6 zeigt bereits in seiner Onlinedokumentation, wo bei diesem Branchenriesen das Hauptaugenmerk liegt. Adobe gibt mit AEM 6 großen und kleineren Entwicklungsfirmen alle Möglichkeiten ihr System anzupassen und zeigt in ihrer Dokumentation in beeindruckender Form, wie und mit welchen technischen Grundhilfsmitteln man responsive Design auf Webseiten entwickeln kann. Leider ist dieser Umfang nur in der Dokumentation geblieben und somit ist das Angebot der Features standardmäßig sehr gering. Die Möglichkeiten die Adobe mit AEM 6 präsentiert sind groß, jedoch bedürfen sie immer eines großen Entwicklungs- und Anpassungsaufwandes von Backendentwicklern.

Die Startseite ist seit AEM 6 in einem neuen Layout vorhanden. Adobe bietet den Nutzern aber in der Version 6 noch die Möglichkeit die Ansicht auf das alte UI umzustellen. Dies gilt auch für alle anderen Funktionen im System, welche mit der neuen Version überarbeitet wurden sind. Optisch ist die Startseite sehr sortiert und übersichtlich.

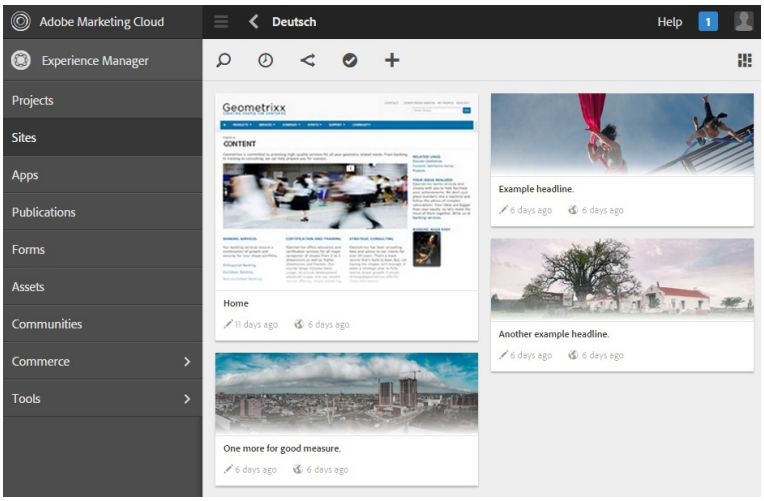

Abbildung 3.12: AEM 6, Startseite

Für Frontendentwicklung bietet Adobe auch mit AEM 6 noch das Repository CRXDE, in welchem alle CSS- und JavaScript-Dateien, sowie die JavaServer Pages (JSP)-Dateien für die Bearbeitung der Templates liegen. Die Konfigurierbarkeit von Media Queries, die bestimmen, welche CSS- und JavaScript-Dateien geladen werden,

findet man, genau wie bei CoreMedia, auch hier nicht. Die dynamische Anpassung der Templates an bestimmte Viewports ist als Standardmechanismus in AEM 6 auch nicht implementiert.

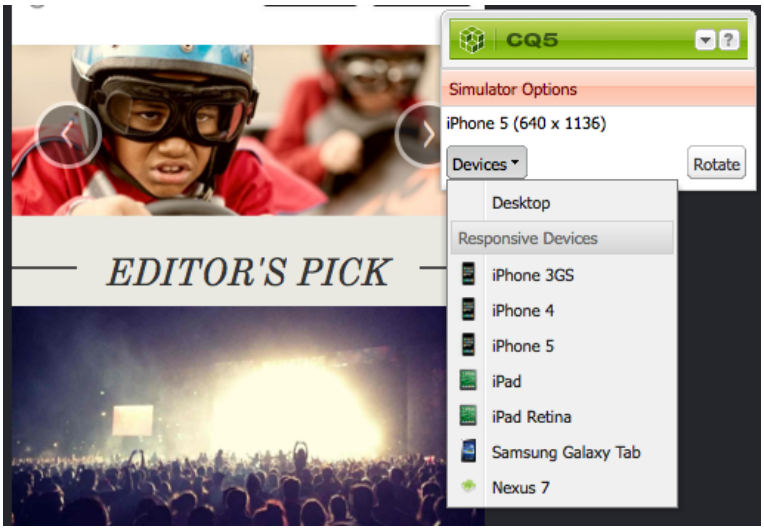

Abbildung 3.13: AEM 6, Previewansicht, [AEM01]

Wie bereits im ersten Abschnitt beschrieben, zeigt AEM 6 beim Preview die Möglichkeiten in Sachen responsive Design für größere Projekte, bei denen auch Backendentwicklung mit betrieben wird. Adobe beinhaltet in ihrem System eine umfangreiche Endgerätebibliothek. Diese ist jedoch nicht standardmäßig implementiert und lässt sich mit Hilfe des CRXDE im Quellcode mandantenspezifisch einbinden. Eine ausführliche und gut verständliche Anleitung ist in der Dokumentation [AEM01] niedergeschrieben. Nach der manuellen Einbindung der Bibliothek, lassen sich alle daran befindlichen Endgeräte der Previewfunktion einzeln hinzu schalten.

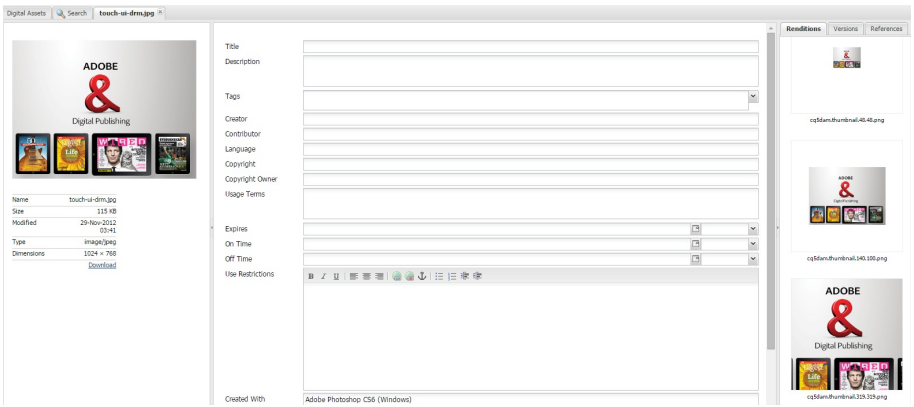

Abbildung 3.14: AEM 6, Renditions

In Projekten, bei denen vertraglich eine Kompatibilität mit bestimmten Endgeräten gewährleistet sein muss, kann dieses Feature optimal genutzt werden und eine große

Unterstützung für beide Vertragsseiten bieten. Da es möglich ist, diese Geräte mit nur wenigen Konfigurationsschritten in die Liste der Previewendgeräte aufzunehmen. Das bedeutet, dass den Frontendentwicklern sowie den Kunden klare Vorgaben und Rahmen für die Entwicklung und Abnahme von Webseiten gesetzt werden können.

In Sachen „Imaging“ verfolgt Adobe hier ein ähnliches Konzept wie CoreMedia 7. Jedes Bild wird in Abhängigkeit von der Komponente, in der es eingepflegt wird, skaliert. Diese Skalierung geschieht beim Hochladen des Bildes und nennt sich bei Adobe „Rendition“. Hierbei werden eine Vielzahl von Thumbnails angelegt, welche das Bild in vielen verschiedenen Auflösungen darstellen. Die verschiedenen Bilder können mittels Media-Query-Einbindung je nach Viewports in die Webseite gerendert werden. Dieses Verhalten muss manuell in das betreffende Template eingebaut werden. Eine ausführliche Beschreibung sowie Codebeispiele findet man in der Onlinedokumentation. [AEM01]

4 Nachweis der Analyseergebnisse

4.1 Prototypische Beispielseite

Nach der Analyse der drei ausgewählten CM Systeme, sollen nun mit Hilfe einer prototypischen Beispielseite die Ergebnisse nachgewiesen werden. Für die praktische Umsetzung soll ein lizenzfreies Design für das bootstrap Framework so detailgetreu wie möglich in allen drei CM Systemen nachgebaut werden. Dabei soll der Fokus sowohl auf der Funktionalität, wie auch auf dem Layout und Design liegen.

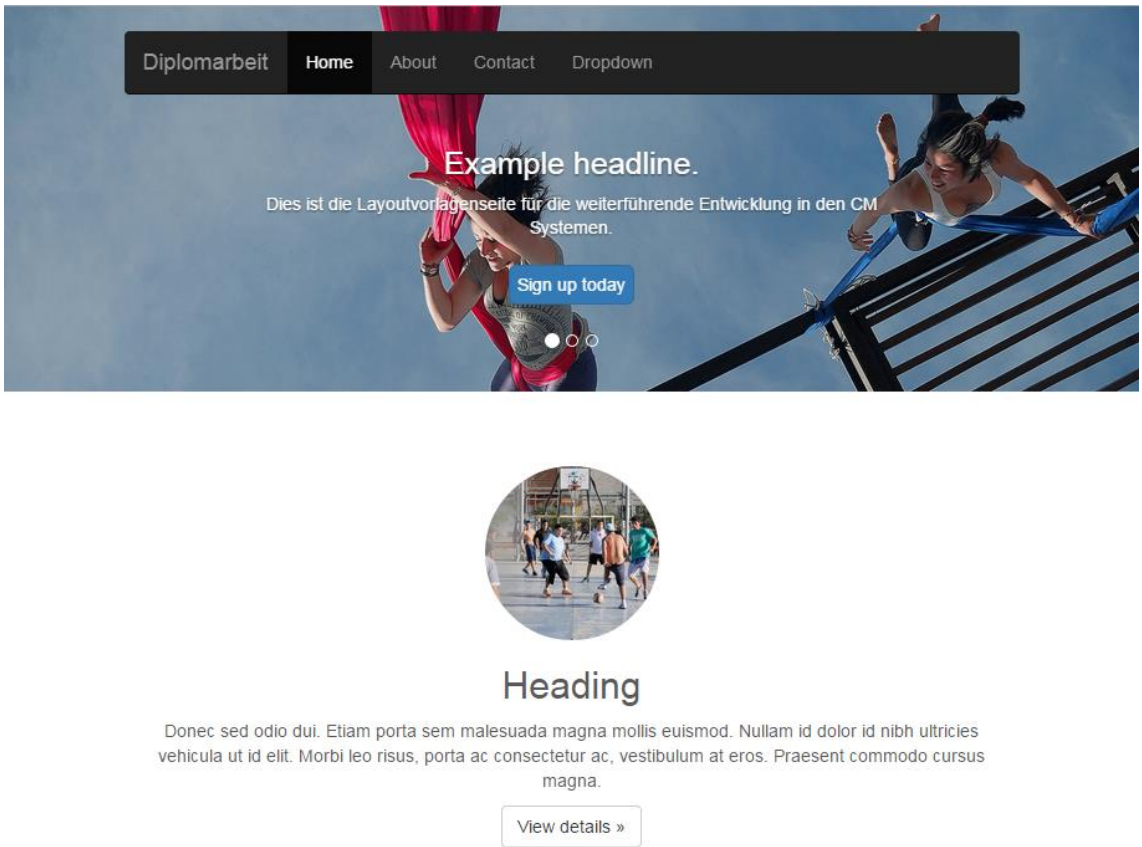

Abbildung 4.1: Prototyp, Oberer Seitenabschnitt

Die prototypische Beispielseite besteht aus fünf Abschnitten, wobei jeder für sich eine responsive Besonderheit in seinem Verhalten aufweist. Der erste Abschnitt ist der Kopf der Seite und beinhaltet die Navigation. Der zweite Abschnitt ist ein Carousel. Das ist in der Fachsprache eine Sliderbox, in welcher Bilder von links nach rechts durch das Bild laufen. Die nächsten zwei Abschnitte sind reine Artikelbeispiele in verschiedenen Darstellungen und immer mit einem Bild versehen. Am Ende der Seite befindet sich der Footerbereich.

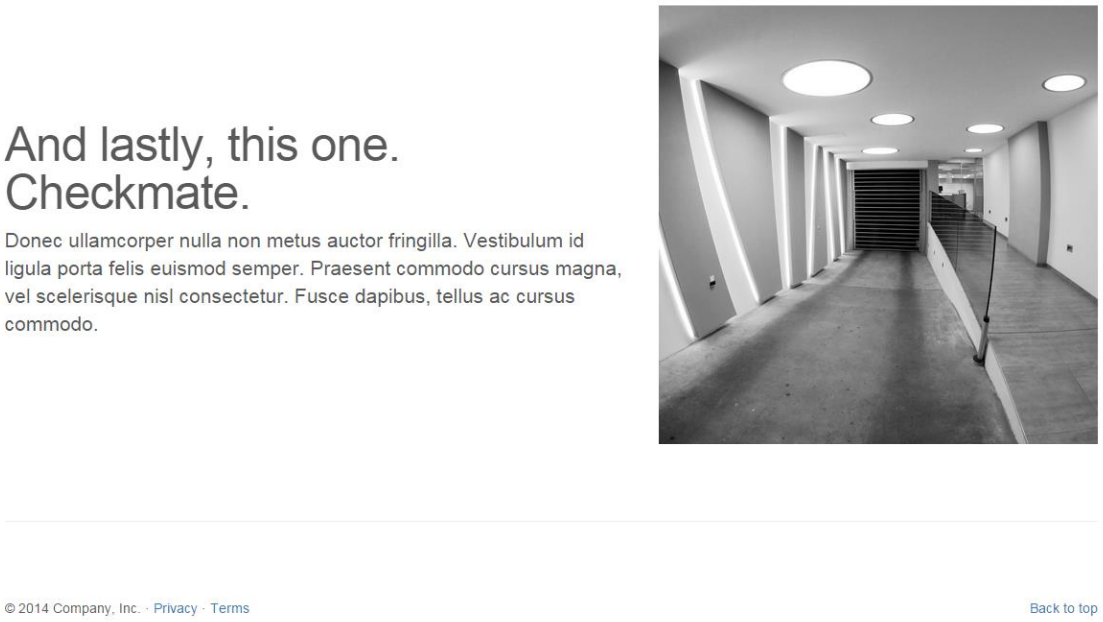

Abbildung 4.2: Prototyp, Unterer Seitenabschnitt

Das responsive Layout der Seite teilt sich in 5 Viewports auf. Der erste Bereich ist für eine Portraitansicht eines Smartphone gedacht. Die Sprungmarke zum nächsten Layout wurde hierbei für 400 Pixel Breite festgesetzt. Der anschließende Bereich umfasst alles zwischen den Breiten von 400 Pixel bis circa 768 Pixel. Dieser Bereich beschreibt die Portraitansicht des nächst größeren Endgerätes - eines Tablets.

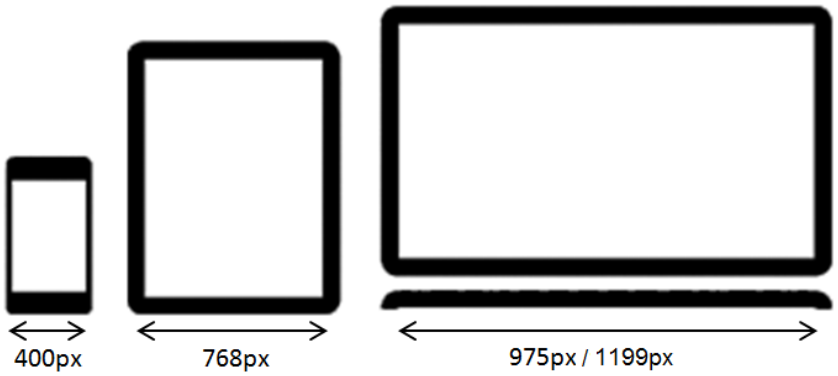

Abbildung 4.3: Prototyp, responsive Design Sprungmarken , [PT01]

Der nächste Viewportwechsel passiert bei einer Breite von 975 Pixel. Diese Sprungmarke definiert das obere Ende von 7” Tablets in der Landscapeansicht.

Die letzte Sprungmarke ist 1199 Pixel. Ab dieser Bildschirmbreite spricht man von eigenständigen Desktopmonitoren oder Laptops mit einer guten Auflösung.

Die Navigationsleiste bildet auf der Beispielseite den Anfang und besteht aus der aktuellen „Home“ Seite und drei weiteren Unterseiten. Diese werden in dem Prototypen nur für die Funktionalität der Navigation hinzugefügt, nicht als eigenständige Seite mit Layout. Für die beiden Viewports der mobilen Endgeräte wird die Navigation in einer Dropdown-Variante dargestellt. Ab der zweiten Sprungmarke wechselt die Navigation zu einer klassischen Leiste mit der Auswahl der Seiten. Das Verhältnis zum nächsten Bereich wechselt, genau wie das Layout der Navigationsleiste, bei der Sprungmarke von 769 Pixel. Auf mobilen Endgeräten wird das Carousel unterhalb der Dropdown-Navigationsleiste angebracht. Das Menü überdeckt beim Aufklappen die Carousel-Komponente. Bei den letzten beiden Viewports bildet das Carousel den Hintergrund der Navigationsleiste und hat diese somit integriert. Die Höhe dieses Abschnittes wächst mit jeder Sprungmarke um die Proportion des Bildes zu bewahren, da sich die Breite immer dynamisch auf den gesamten Bildschirm ausbreitet.

Im dritten Bereich der Seite befinden sich kleine Teaser, welche auf einen größeren Artikel verweisen sollen. Sie bestehen aus einem zentrierten Bild mit abgerundeten Ecken, sowie einen Text mit Überschrift und Link zum Weiterlesen. Die drei Teaser werden auf den ersten drei Viewports aufgrund von Übersichtlichkeit und Platzmangel untereinander dargestellt. Wenn die Breite den Wert von 1199 Pixel überschreitet, werden die Teaser nebeneinander gleich verteilt dargestellt.

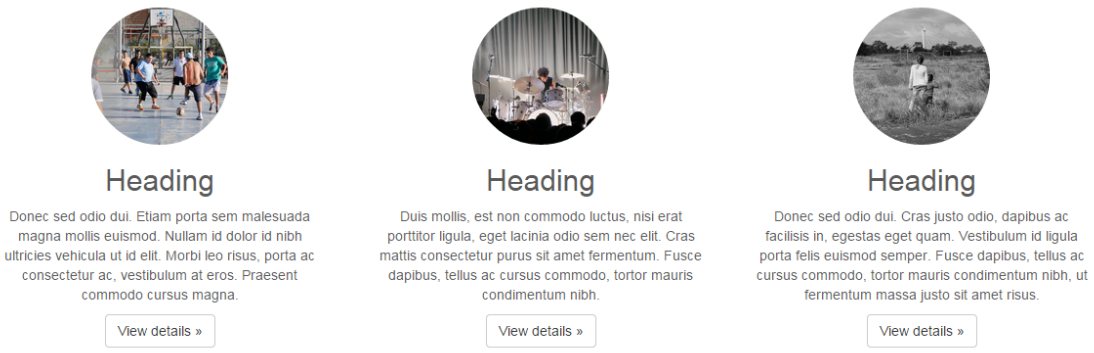

Abbildung 4.4: Prototyp, Teaser über einer Breite von 1199 Pixel

Der vorletzte Abschnitt der prototypischen Beispielseite beinhaltet drei Artikel mit einer Überschrift und einem Bild. Das Layout wird hierbei an der Sprungmarke von 975 Pixel verändert. Bei allen Bildschirmbreiten, die diesen Wert unterschreiten, wird das Bild auf fast die gesamte Bildschirmbreite dargestellt und passt sich dynamisch auf diese an. Die Überschrift und der Text stehen entweder über oder unter dem Bild. Nach dem Layoutwechsel werden diese links oder rechts neben dem Bild dargestellt. Wobei das Bild nun nur noch einen kleinen prozentualen Teil der Bildschirmbreite in Anspruch

nimmt. Die Position der Texte zum Bild verhält sich alternierend. Die Schriftgrößen der Überschrift, sowie der Text passen sich ebenfalls in Abhängigkeit zum Viewport an.

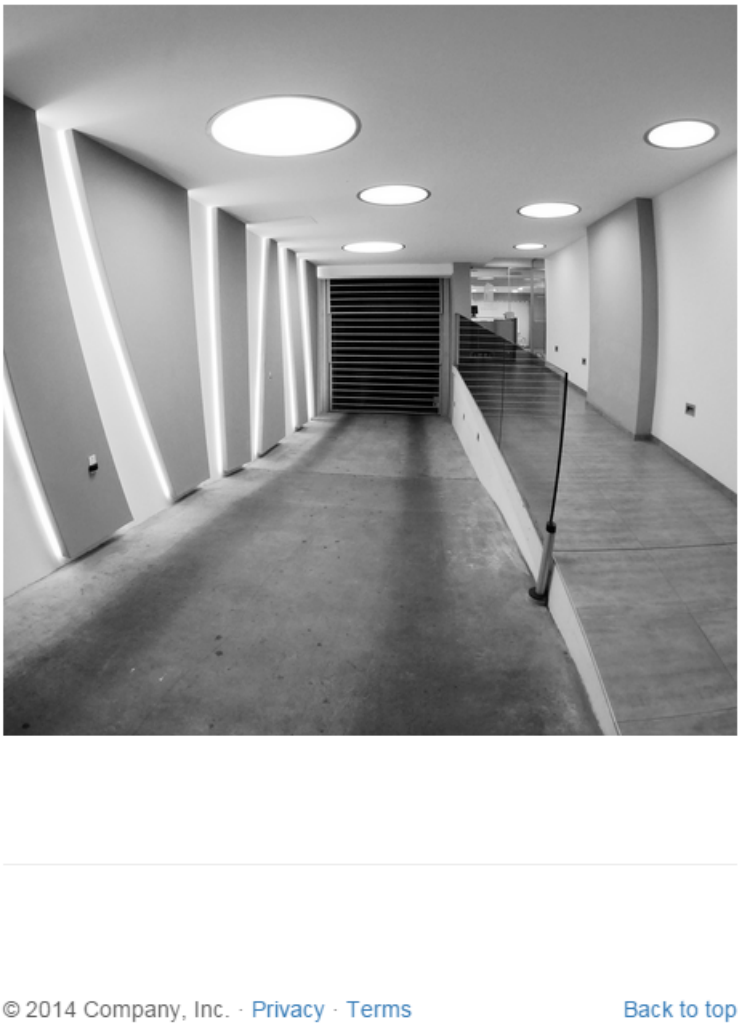

Abbildung 4.5: Prototyp, Artikel und Footer

Den Abschluss einer jeden Seite bildet der Footerbereich. Dieser beinhaltet das Copyrightlogo und drei Links. Alle Links sind nur Dummylinks und haben die „Home“ Seite als Ziel. Sie dienen lediglich zur Umsetzung des Layouts und zum verdeutlichen des Verhaltens. Der „Back to top“ Link wird immer rechtsbündig auf der Seite dargestellt. Die anderen beiden Links werden mit einem festen Abstand zum Copyrightlogo, mit einem Punkt getrennt, daneben angezeigt.

Die in diesem Kapitel beschriebene prototypische Beispielseite wird im Zuge dieser Diplomarbeit versuchsweise in die drei ausgewählten CM Systeme integriert. Das Ziel ist eine Nutzung und Bewertung der Standardmechanismen der jeweiligen CM Systeme und eine allgemeine Bewertung der Umsetzung von responsive Webseiten mit einem vordefinierten Layout.

Ein weiteres Ziel der Umsetzung sollte es sein, eine Redaktionierbarkeit des Inhalts zu gewährleisten. Das bedeutet, dass es vermieden werden sollte, jegliche Inhalte statisch zu generieren. Eine Anpassung der Templates-, JavaScripts- und CSS-Dateien sollte dabei als Unterstützung für die Umsetzung genutzt werden.

4.2 Magnolia 5

Magnolia liefert, wie auch die beiden anderen CM Systeme, standardmäßig ein mit Demoinhalt gefülltes Default-Template aus. Dieses vordefinierte Template heißt bei Magnolia „demo-project“ und bietet die Arbeitsgrundlage für die Umsetzung der prototypischen Beispielseite in diesem CMS.

Das „demo-project“ Template besitzt einige vordefinierte Komponenten, welche zum Großteil nicht bei der Umsetzung für diese Diplomarbeit benötigt werden. Im Kopfbereich der Seite befindet sich ein Logo, eine Suche sowie diverse Metalinks. Die Navigation, wie auch das standardmäßig redaktionierte Carousel, wird für die Umsetzung der Beispielseite benötigt.

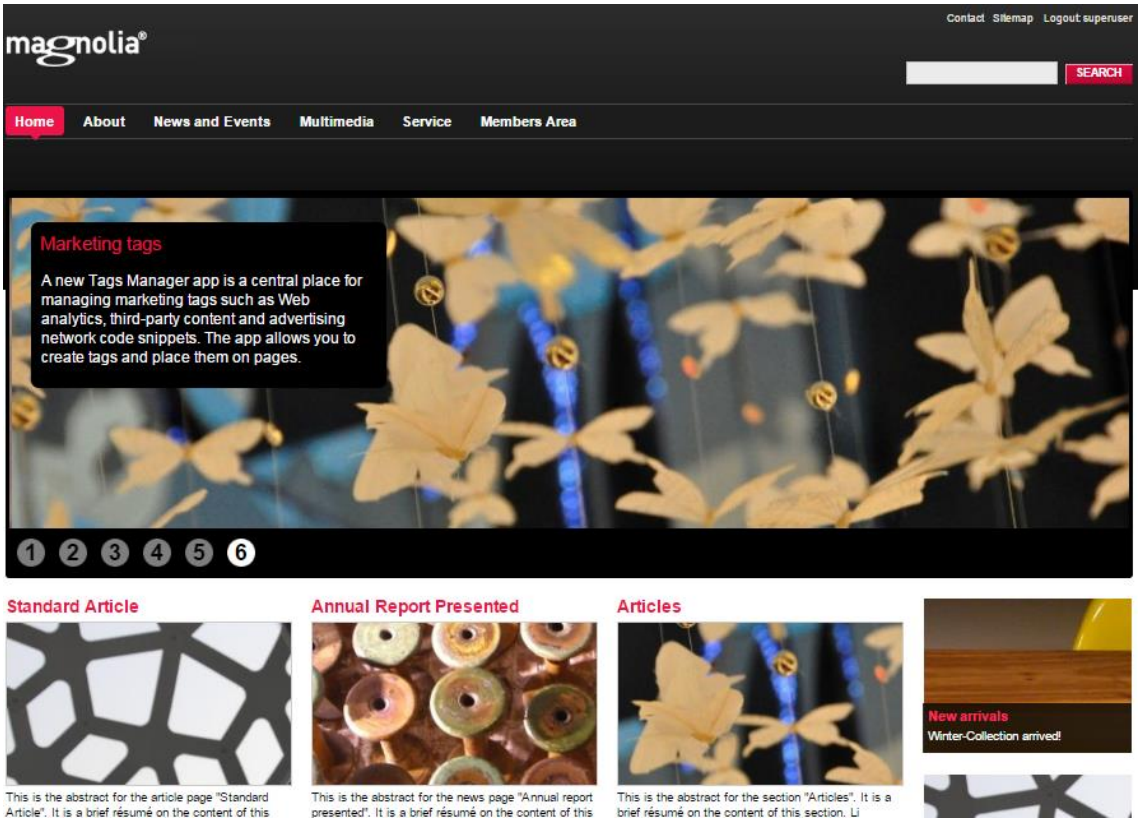

Abbildung 4.6: Magnolia 5, „demo-project“ Template, unterer Bereich

Die Hauptseite ist im Template in einen Haupt- und einen Seitenbereich geteilt. Die Elemente, welche sich in der rechten Spalte befinden, lassen sich problemlos mit

dem Edit-Modus von Magnolia entfernen. Die Aufteilung der Seite kann mittels CSS angepasst werden. Im Hauptbereich befinden sich drei nebeneinander stehende Artikel, welche bereits den Vorgaben durch den Prototypen entsprechen.

Die unterhalb der Artikel verweisenden Links sind einer eigenständigen Komponente untergeordnet und können als Ganzes von der Seite mittels des Edit-Modus entfernt werden. Das Gleiche passiert mit dem anschließenden Carousel für Teaser. Der Footerbereich der Seite lässt sich, bis auf das Copyrightlogo, komplett redaktionell anpassen. Es ist eine Standardmenge an Spalten voreingestellt, welche erweitert, wie auch vermindert werden kann. Alle im Footerbereich zu sehenden Links sind mit Hilfe des Edit-Modus veränderbar.

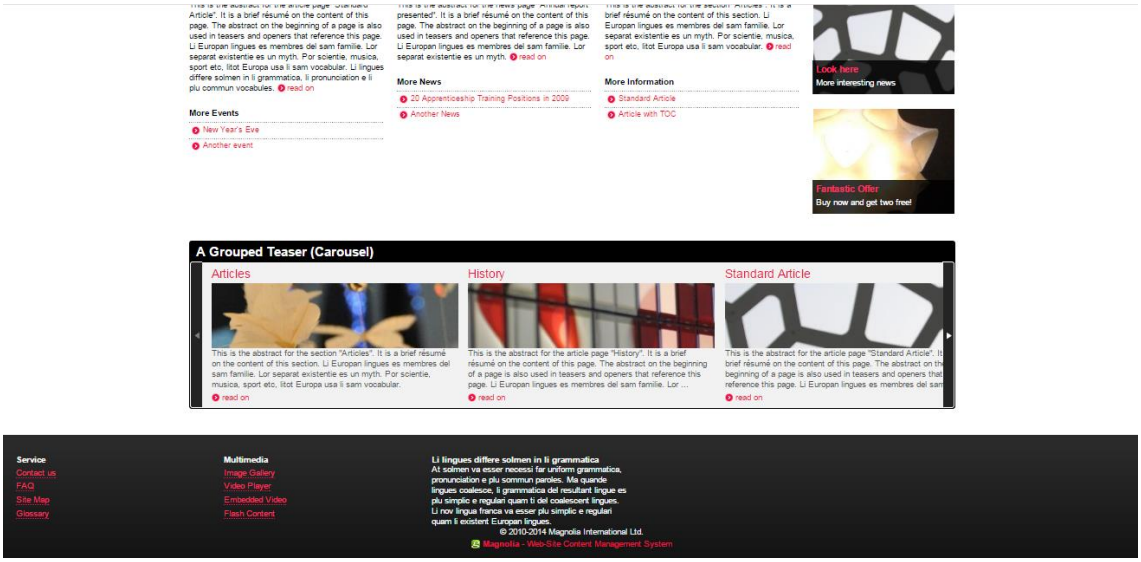

Abbildung 4.7: Magnolia 5, „demo-project“ Template, oberer Bereich

Als ersten Schritt bei der Umsetzung der prototypischen Beispielseite, werden alle, soweit es sich mit dem CMS realisieren lässt, benötigten Komponenten redaktioniert. Der Kopfbereich und die darin integrierten Elemente, wie die Suche, die Metanavigation, sowie das Magnolia Logo, können in der „Seitendefinition“ von Magnolia angepasst oder entfernt werden. Die anderen nicht benötigten Komponenten können im Edit-Modus entfernt werden.

Für die Umsetzung des responsive Menüs wird ein lizenzfreies Plug-In verwendet [CMS05]. Das Plug-in beinhaltet neben den benötigten Bildern noch eine JavaScript- und eine CSS-Datei. Für die Einbindung des Plug-Ins muss der Aufbau der Navigation angepasst werden. Diese Anpassung wird in dem Freemarker-Template „contentNavigation“ durchgeführt. Wichtig bei der Struktur der Navigation ist ein „div“-Element mit der CSS-Klasse „rmm“, welches ein „ul“ Element umschließt. Die CSS-Klasse spielt hierbei die entscheidende Rolle, da die JavaScript-Datei des Plug-Ins auf diese Klasse reagiert. Bei Magnolia muss lediglich dem „div“ Element

die erforderliche CSS-Klasse zugewiesen werden. Der Aufbau der Navigation ist dem des Plug-Ins identisch. Für eine detaillierte optische, sowie funktionelle Anpassung des Menüs an das prototypische Vorbild, bedarf es noch zweier JavaScript-Anpassungen. Zum Einen ist die maximale Breite der Navigationsleiste fest an das „div“ Element geschrieben. Zum anderen fehlt der „Diplomarbeit“ Textzug. Der Textzug lässt sich als erstes Listenelement in die Navigationsleiste einfügen. Die festgeschriebene maximale Breite der Navigationsleiste bekommt einen leeren Wert zugewiesen und ist somit nicht mehr relevant. Nach dieser Anpassung kann mittels CSS die Breite der Navigationsbar variabel festgelegt werden.

Bei einer Anpassung der Standard-Template-Dateien ist es bei Magnolia notwendig, den Template mitzuteilen, dass es die neue Struktur verwenden soll. Wird dies nicht explizit definiert, werden die Anpassungen nicht auf den Publisher übertragen. Diese Definition kann durch einen Haken an der Checkbox „Enable template“ direkt in der Template-Datei vollzogen werden.

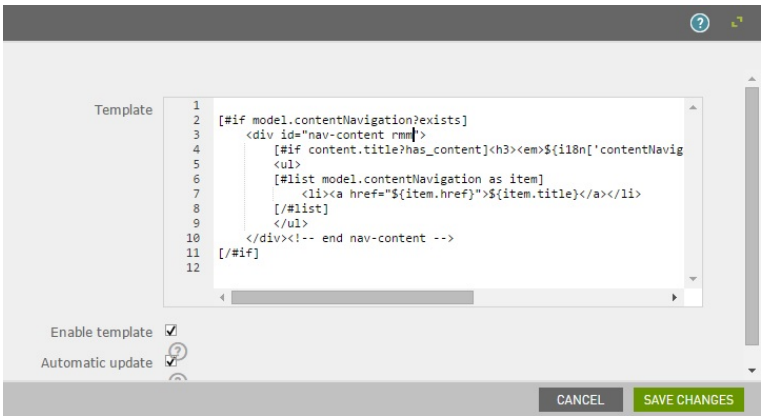

Abbildung 4.8: Magnolia 5, Templateanpassung

Magnolia fügt bei der Navigation die „Home“ Seite als einen separaten Navigationspunkt hinzu, auch wenn diese als eine Überseite in der Struktur definiert ist. Des weiteren vergibt Magnolia dem aktiven Seitenelement in der Navigation eine separate CSS-Klasse. Das ist bei der Umsetzung der prototypischen Beispielseite von Vorteil, um am Ende die aktive Seite mittels CSS optisch hervorzuheben. Für die Implementierung werden auf der Beispielseite, bis auf die benötigten drei „About“, „Contact“ und „Dropdown“, alle Unterseiten entfernt.

Die Carousel-Komponente bietet vom allgemeinen Aufbau eine gute Grundlage zum Übertragen der Komponente aus der prototypischen Beispielseite. Lediglich der Linkbutton mit den dynamischen Texten, wird von Magnolia nicht mit angeboten. Hierfür erfolgt eine Dialoganpassung und die Anpassung des Templates für die Carousel-Komponente. Bei der Anpassung des Dialogs muss unter „Konfiguration“ ein zusätzliches Feld unter dem entsprechenden Tab angelegt werden. Der vergebene Feldname lässt sich anschließend in dem Template dynamisch auslesen. Somit kann

man für jedes Carousel-Item den Text des Buttons redaktionieren und als Folge dessen, das Ganze dynamisch für den Redakteur gestalten.

Das Carousel arbeitet bei Magnolia nicht standardmäßig responsive. Magnolia rendert für das Carousel alle Bilder, die der Reihe nach angezeigt werden, nebeneinander und erzeugt somit ein sehr breites Bild. Wenn sich dieses Bild mit Hilfe der CSS-Anweisung nun auf 100% strecken soll, werden alle Bilder nebeneinander auf der vollen Breite angezeigt. Da die Breite der Bilder jedoch anhand von JavaScript fest an die Elemente geschrieben wird, muss dieses Problem ebenfalls mittels JavaScript, gelöst werden. Dem Bild muss also eine prozentuale Breite in Abhängigkeit von der Anzahl der Bilder zugewiesen werden. Anschließend wird das einzelne Bild auf die volle Breite des Viewports gesetzt. Die mittels JavaScript vergebene feste Breite des darüber liegenden „div“ Elementes wird ebenfalls mit Hilfe von JavaScript auf 100% gesetzt. Dies hat immer eine Ausbreitung des „div“ Elementes, wie auch der darunter liegenden Bilder, auf die volle Bildschirmbreite zur Folge.

Abbildung 4.9: Magnolia 5, responsive Carousel

Das fehlende responsive Verhalten der Carousel Komponente lässt sich mit Hilfe des oben gezeigten Verfahrens erstellen und somit das Layout und Verhalten der prototypischen Beispielseite erzeugen. Es hat dennoch einen starken Einfluss auf das Endresultat dieser Arbeit. Aufgrund des fehlenden responsive Verhaltens wurde diese Komponente in Magnolia auf allen mobilen Geräten deaktiviert und lässt sich somit nur mit Hilfe einer Simulation im Browser testen.

Das Kontrollelement des Carousels besteht bei Magnolia aus einer Liste gefüllt mit „button“ Elementen, welche nochmals ein „span“ Element mit einer fortlaufenden generierten Zahl beinhaltet. Da die Zahlen in unserem Falle nicht nötig sind, jedoch mittels JavaScript und nicht im Template generiert werden, blendet man die „span“ Elemente mit Hilfe von CSS einfach aus.

Nach der Carousel-Komponente wird auf der Seite eine Spaltenkomponente dargestellt. Diese teilt den Bereich in drei gleich große Teile, in denen wiederum jeweils ein Artikel eingebunden ist. Dieser Aufbau entspricht, bis auf den statischen Linktext, dem des Prototypen. Der statische Linktext „read more“ wird im Template der Artikelkomponente durch den Text „View details“ ersetzt.

Unterhalb der Artikel sollen drei sogenannte „Text und Bild“ Komponenten eingebaut werden. Standardmäßig lässt sich diese Komponente nicht in den gewünschten Bereich einfügen. Diese muss vorab unter der „Seitendefinition“ aktiviert werden. Dazu wird lediglich eine Verknüpfung unterhalb der „base Area“, der aktuelle Bereich auf der Hauptseite, zu der „stkTextImage“ Komponente gesetzt. Anschließend lassen sich die drei Komponenten problemlos auf die Seite redaktionieren.

Die Umsetzung der alternierenden Positionierung der Bilder zu den Texten benötigt eine weitere Templateanpassung. Im mobilen Design werden die Bilder bei jedem zweiten Teaser über, ansonsten unter, dem Text dargestellt. Es ist möglich, dies mit zwei verschiedenen Methoden zu realisieren. Für die erste Methode nutzt man reines CSS. Mit Hilfe der Flexbox-Funktion lassen sich „div“ Elemente in ihrer Reihenfolge nur mit CSS verändern und somit komplett variabel verschieben. Bei Magnolia entschied man sich für die zweite Variante, da die Komponente bereits eine Auswahlmöglichkeit in ihren Einstellung besitzt, wo das Bild positioniert werden soll. Diese wird beim Rendern der Komponente abgefragt und in Abhängigkeit dieser die Position der Elemente festgelegt. Bei einer Auswahl „links“ werden beispielsweise erst die Überschrift und der Text gerendert und anschließend das Bild. Genau umgekehrt verhält es sich bei der Auswahlmöglichkeit „rechts“.

Im Footerbereich der Seite wurden alle Links entfernt und das Copyright durch den eigenen Schriftzug ersetzt. All diese Anpassungen konnten auf rein redaktioneller Basis durchgeführt werden. Die drei benötigten Links im Footerbereich wurden mit Hilfe von zwei separaten Linklisten erstellt. Da bei der Implementierung des Prototypen keinerlei Linklist-Überschriften benötigt wurden, konnte dieser Dialogeintrag für das Setzen der CSS Klassennamen genutzt werden. Dabei wurde im Template die Überschrift entfernt und der auszugebende Wert als CSS-Klasse an ein umrandendes „div“ Element geschrieben. Somit lassen sich die Links einzeln und unabhängig voneinander im Footerbereich positionieren.

Bei der Umsetzung des gewünschten Layouts waren die internen CSS-Einbindung mit Hilfe von Media Queries eine gute Unterstützung. Aufgrund der in Magnolia integrierten Definitionsmöglichkeit der Media Queries, lassen sich die CSS-Befehle strukturieren und machen eine Bearbeitung und Verwaltung der einzelnen Layoutstufen, für den Redakteur, sehr praktikabel. Die „Imaging“ Funktion bei Magnolia ist eine tolle Sache um die Performance der Seite zu optimieren. Es muss jedoch beachtet werden, dass einige Bilder von bestimmten Komponenten diese Funktion schon standardmäßig nutzten. Die Artikelbilder haben beispielsweise eine Auflösung von 240x240 Pixel auf mobilen Geräten. Werden diese nun auf die gesamte Bildschirmbreite gestreckt, wird das Bild sehr verpixelt dargestellt. Das „Imaging“ lässt sich in der „Seitendefinition“ deaktivieren, wie auch editieren.

Zusammenfassung der Anpassung am System:

- Navigation (Template) - CSS-Klasse
- CarouselItem (Template) - Kontrollpanel und Skalierung
- Artikel (Template) - Anordnung Bild zu Text
- Teaser (Template) - Linktext statisch
- Footer (Template) - Copyrighttext
- Imagekomponte (Dialog) - Linktext dynamisch

4.3 CoreMedia Blueprint

Bereits auf den mitgelieferten Beispielseiten lässt sich CoreMedias abweichendes Radaktionsprinzips erkennen. Dieses trägt den Namen „Corporation-en“ und ist wie eine technische Produktseite aufgebaut. Im Kopfbereich der Seite befinden sich, wie schon bei Magnolia, eine Suche, ein Logo, diverse Meta-Links und, anders als bei Magnolia, noch ein Login-Verweis. Die Hauptseite ist in drei große Spalten aufgeteilt, welche nochmals unterschiedliche inhaltliche Abschnitte ausweisen.

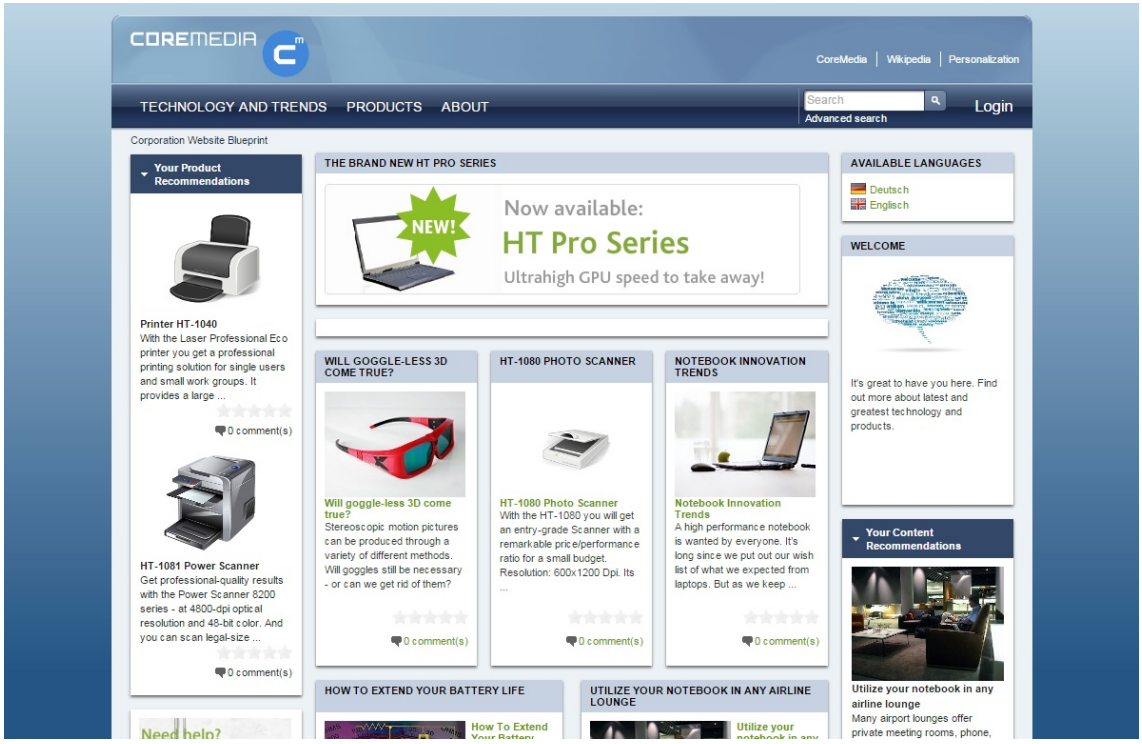

Abbildung 4.10: CoreMedia, Demoseite

Die linke und rechte Spalte ist mit Referenzen auf Artikel versehen und wird nicht für die Umsetzung der prototypischen Beispielseite genutzt. Der mittlere Bereich ähnelt von der Anordnung der Artikel der Beispielseite. Der Hauptbereich startet mit einem Carousel.

Dieses beinhaltet verschiedene Produkte als Verweis. Dem Carousel schließt sich eine, in drei Spalten aufgeteilte, Artikelreferenz an. Diese beiden Komponenten können für die Implementierung genutzt, müssen jedoch noch im Aufbau und Aussehen an den Prototypen angeglichen werden.

Der untere Bereich der Seite ist, wie der obere, in eine rechte, eine linke und eine mittlere Spalte geteilt. Die beiden äußeren Spalten können weiterhin vernachlässigt werden. In der Mitte folgt nach der Drei-Spalten-Komponente eine weitere Zwei-Spalten-Komponente, welche ebenfalls nicht für die Implementierung benötigt wird.

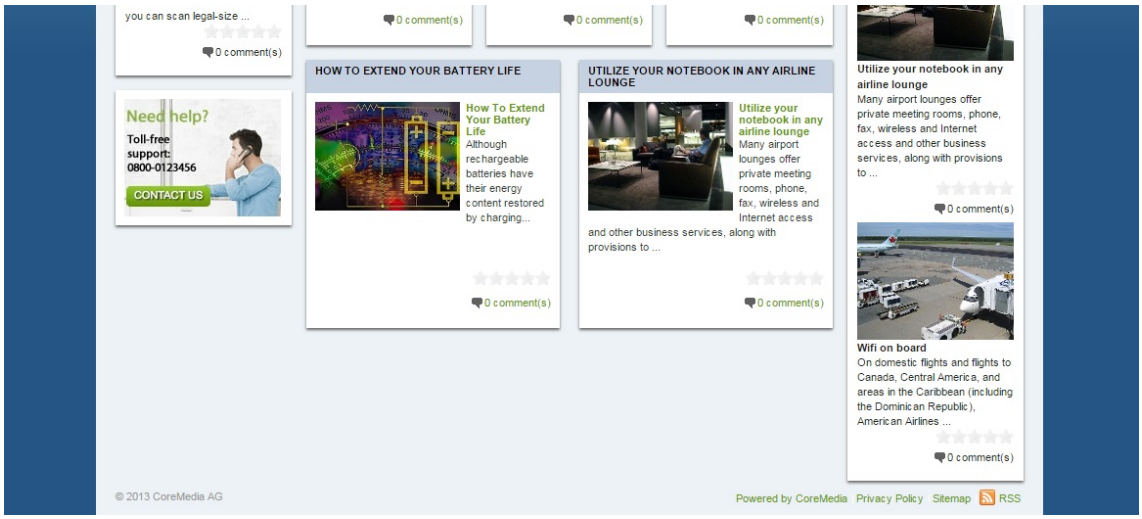

Abbildung 4.11: CoreMedia, Demoseite

Der Fußbereich weist auf der linken Seite den generellen Copyright-Schriftzug auf. Die rechte Seite des Fußbereiches ist mit Links zu „Powered by CoreMedia“, „Privacy Policy“ und „Sitemap“ bestückt, sowie einen Verweis auf ein „RSS-Feed“. All diese Links sind redaktionell in die Seite eingebaut und können auf diese Weise entfernt werden.

Allgemein lässt sich bei der Beispielseite von CoreMedia auch eine weitere Funktion feststellen, welche die anderen beiden CM Systeme nicht aufweisen. Alle Teaser und Artikel haben eine Bewertungs- und Kommentarfunktion. Die Artikel und Teaser sind bei CoreMedia aus redaktioneller Sicht ein und dieselbe Komponente. Lediglich in Abhängigkeit von ihrem Elternobjekt wird definiert, ob sie als Artikel oder als Teaser dargestellt werden sollen. In der Komponente sind dafür jeweils zwei verschiedene Felder für Überschrift und Text vorgesehen. Die Komponente kann folglich, je nach Elternobjekt, verschiedene Inhalte ausgeben.

CoreMedia verwendet „Views“ für ihre Templates. Es handelt sich dabei um JSP-Dateien. Diese werden mit Hilfe der „Views“ noch detaillierter strukturiert. Die Dateinamen werden als eine Art Gliederung genutzt. Mit der untenstehenden

Anweisung wird der „View“ „asAccordionItem“ in das Template eingebunden. Es wird nun im Dateinamen nach „.asAccordionItem“ gesucht und der Inhalt der JSP-Datei in das Template eingebunden.

```
<cm:include self="${item}" view="asAccordionItem"/>
```

Kommentiert man diese „Include“ Anweisungen aus, kann man ganze Teile eines Templates ausschalten und bei Bedarf wieder hinzufügen. Mit Hilfe dieser Strukturierung lassen sich Templates sehr übersichtlich aufbauen und verwalten. Als großen Nachteil beim Nachbau des Prototypen in CoreMedia erwies sich, dass die JSP-Dateien in Bibliotheken abgelegt werden und nicht direkt im System zu editieren sind. Das hat bei jeder Änderung an der Templatedateien einen Neustart des Systems als Konsequenz. Alle für die Implementierung notwendigen JSP-Dateien lassen sich in der Bibliothek „cae-base-lib-26.jar“ finden.

Die erste Templateanpassung wird im Headerbereich der Seite notwendig. Die in der Navigation eingebundene Suche und der Login-Bereich lassen sich, wie auch das CoreMedia Logo, nicht mittels der Redaktionsoberfläche anpassen. Diese müssen, anders als alle in der Metanavigation auftauchenden Links, in der JSP-Datei auskommentiert werden. Die Links können in der Hauptseite des Mandanten angepasst und entfernt werden. Dasselbe gilt auch für die im Footerbereich auftauchenden Links.

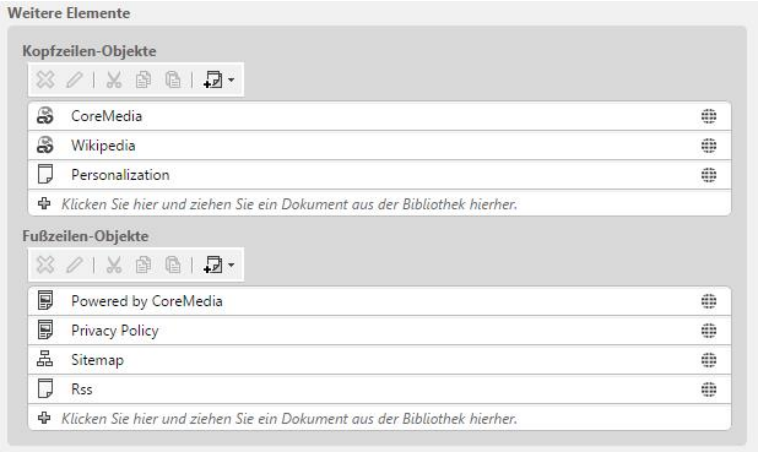

Abbildung 4.12: CoreMedia, Footer- und Headerlinks

Für die Nutzung des responsive Menü Plug-Ins muss, wie auch schon bei Magnolia, dem die „ul“ Liste umschließendem „div“ Element die CSS-Klasse „rmm“ angeheftet werden. Es handelt sich hierbei um eine statische Anpassung der „CMChannel.asTopNavigation.jsp“-Datei. Nach der Einbindung der mitgelieferten CSS- und JavaScript-Datei, funktioniert das responsive Menü bereits in der gewünschten Form. Es bedarf nur noch der Anpassung an das prototypische Layout und die Sprunggrößen der Navigationsleiste.

Auf der Standardseite ist, wie im oberen Abschnitt beschrieben, eine Carousel-Komponente eingebaut, welche für die Umsetzung des Prototypen benötigt wird. Inhaltlich werden die verlinkten Teaser bearbeitet und die Texte und Bilder nach Vorbild des Prototypen angepasst. Damit sich die Komponente responsive verhält, bedarf es einer Änderungen an der komponentenspezifischen JavaScript-Datei. Hierbei wurden den einzelnen Bildern feste Breitenwerte zugewiesen. Diese werden durch den Wert „100%“ ersetzt. Des Weiteren wird das Bild mit einer dynamischen Höhe von „100%“ versehen. Dies hat, wie schon bei Magnolia, den Vorteil, dass man die Maße des Carousels mit Hilfe des darüberliegenden „div“ Elementes via CSS steuern kann. An der JSP, die für das Rendern der einzelnen Carousel Elemente verantwortlich ist, bedarf es noch einer Anpassung des Linkbuttons. Der Inhalt dieses Buttons wird aus der Überschrift des Ziel Teasers generiert und kann somit dynamisch und itemspezifisch editiert werden.

Auch die Kontrollleiste der Carousel Komponente benötigt eine Anpassung des Codes. Bei CoreMedia werden unterhalb der Listenelemente Links verwendet. Diese Links haben ein Bild eines grauen Kreises eingebunden. Macht man dieses Bild mittels CSS unsichtbar, verschwindet auch der Link und die Steuerung ist nicht mehr nutzbar. Das Bild wird im Template durch ein leeres Dummybild ersetzt, welches auf die gewünschte Größe skaliert werden kann. Anschließend werden die Kreise des prototypischen Layouts direkt mit CSS an den Listenelementen deklariert.

Nachdem die Funktionalität und das Design der Carousel Komponente an den Prototypen angeglichen wurde, folgt die Anpassung der Drei-Spalten-Komponente. Die bereits im Standardinhalt eingebunden Teaser werden als redaktionelle Grundlage genutzt. Die Bilder und Texte können direkt in den Komponenten verwaltet werden. Die Kommentar- und Bewertungsfunktion wird in den JSP-Dateien deaktiviert, da sie für die Umsetzung nicht benötigt werden. Auch die doppelten Überschriften oberhalb des Bildes werden entfernt.

Den Teasern soll weiterführend noch der Linkbutton „View details“ hinzugefügt werden. CoreMedia bietet hierfür keine redaktionelle Möglichkeit, womit sich eine Umsetzung in der JSP-Datei nicht vermeiden lässt. Das „View“-Model von CoreMedia erschwert diese Umsetzung. Die Artikel und Teaser sind zwar unterschiedliche JSP-Dateien, jedoch binden sie eine Reihe von gleichen JSP-Dateien ein. Erweitert man die am Ende dieser „Includekette“ stehende „content“-Datei um den Linkbutton, wird dieser auch in den Artikeln angezeigt. Es ist möglich diesen Linkbutton mittels CSS auszublenden. Da dies keine saubere Umsetzung wäre, wird auf die folgende Methode zurückgegriffen.

Um den Linktext nur bei Teasern anzeigen zu lassen, dürfen die beiden Komponenten nicht mehr die gleichen JSP-Dateien einbinden. Die Drei-Spalten-Komponenten und der Hauptbereich der Seite binden beide die „CMTeasable.asTeaser“-Datei ein. Diese wird dupliziert. Somit können beide Eltern-Dateien nun verschiedene „Views“ einbinden.

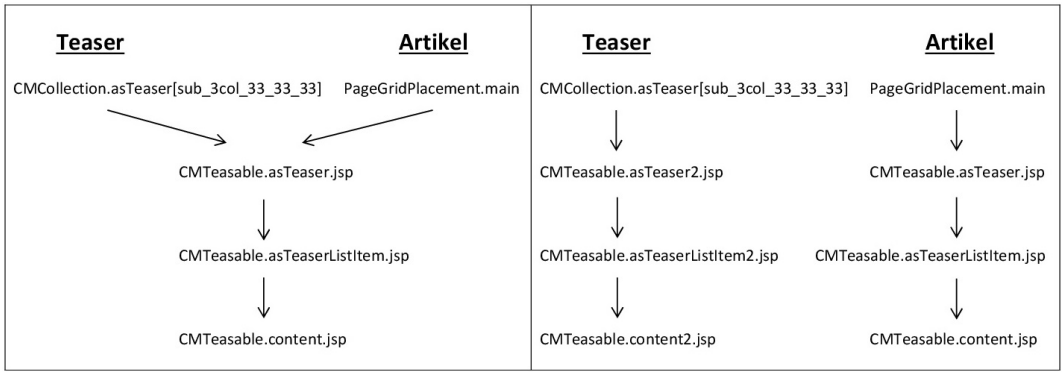

Abbildung 4.13: CoreMedia, Viewstruktur des Artikels und des Teasers, alte Struktur (links) und neue Struktur (rechts)

Dasselbe muss anschließend noch mit den „Views“ von „CMTeasable.asTeaserListItem“ und „CMTeasable.content“ erfolgen. Im Letzteren wird in einen der beiden „Views“ der Linkbutton erzeugt, in den anderen nicht.

Für die responsive Umsetzung des Layouts mit CSS bekommt das Eltern „div“ der Drei-Spalten-Komponente, ab der Bildschirmbreite 1199 Pixel, die Eigenschaft „flex“. Damit lassen sich die Spalten nebeneinander darstellen. Den einzelnen Spalten werden noch die gleichen Indizes für das „flex“ Verhalten gegeben. Somit teilen sich die drei Spalten gleich groß, auf die verfügbare Breite des „div“ Elements auf.

Für die im Layout folgenden Artikel müssen die gleichen Templateänderungen, wie schon bei den Teasern, vorgenommen werden. Da die Komponente in diesem Fall nicht unter einem Spaltenelement eingebunden ist, sondern direkt in dem Hauptbereich der Seite, wird nun der neu erzeugte Artikel- und nicht der Teaserstrang genutzt. Aufgrund der Duplizierung der JSP-Dateien, nach dem Anpassen des Teasers, sollten die Kommentar- und Bewertungsfunktion, sowie die doppelte Überschrift, bereits entfernt sein. Der Inhalt wird mit Hilfe des CM Systems angepasst.

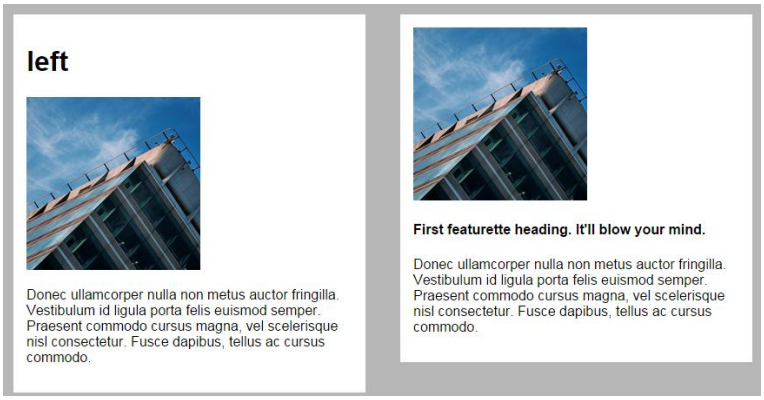

Abbildung 4.14: CoreMedia, Teaser (links) und Artikel (rechts)

Aufgrund der alternierenden Anordnung der Bilder zu den Texten in den Artikeln muss hierfür eine weitere JSP-Anpassung vorgenommen werden. Da es sich bei dieser Diplomarbeit um eine prototypische Implementierung im Frontendbereich handelt, wird auf Anpassung von Javaklassen verzichtet. Aus diesem Grund wird für die CSS-Klassendeklaration in diesem Fall die Teaserüberschrift genutzt. Sollten die in der Arbeit erstellten Artikel auf der Seite nochmals als Teaser eingebunden werden, entspricht die Überschrift des Artikel der CSS-Klasse. Generell wird empfohlen, bei allen relevanten Komponenten, ein extra CSS-Klassenfeld hinzuzufügen, um somit redaktionell diesen CSS-Klassen für die Erstellung von Customlayouts zuzuweisen. Die Anordnung der Bilder und Texte wird nach Vergabe der CSS-Klassen mit Hilfe der „flex“-Funktion im CSS-Code realisiert.

Im Fußbereich der Seite wurden bereits alle nicht benötigten Links im CM System entfernt. Die Anpassung des Copyright-Schriftzuges wird, wie bei Magnolia, direkt in der JSP geändert. Die gewünschten drei Links des Footerbereiches werden ebenfalls direkt im Template implementiert, da sie auf allen Seiten des Mandanten gewünscht sind und nicht händig im System entfernt werden sollen.

Für die Realisierung des mobilen Layouts, welches die mobile Navigationsleiste verwendet, ist eine Änderung der entsprechenden Templates notwendig. Ziel soll es sein, dass sich die ausgeklappte Navigationsleiste über das Carousel Element ausbreitet. Da sich der Hauptbereich der Seite hingegen relativ zur Größe der Navigationsleiste positioniert, schiebt sich dieser immer automatisch mit nach unten. Setzt man den Inhaltsbereich der Seite nun absolut zur Navigation, verschiebt sich der Fußbereich der Seite relativ zur Navigationsleiste. Um dieses Verhalten zu vermeiden, wird dem Inhalts- und Fußbereich der Seite ein gemeinsames „div“ Element als Elternobjekt zugewiesen. Dies kann anschließend absolut zur Navigationsleiste gesetzt werden, ohne dass sich das gesamte Layout der Seite verschiebt.

Wie schon bei Magnolia , muss auch bei CoreMedia der Navigationsleiste ein Schriftzug „Diplomarbeit“ hinzugefügt werden. Auch der Wert der festen maximalen Breite dieser muss gelöscht werden, um die Breite der Leiste mit CSS dynamisch anzupassen. Die Integration der CSS- und JavaScript-Dateien erfolgt, wie bereits beschrieben, in der Hauptseite des Mandanten und wird auf alle Unterseiten vererbt. Die Anpassung dieser lässt sich an Hand eines Textfeldes direkt im CMS editieren.

Eine Funktion zur Performanceoptimierung, wie das „Imaging“ bei Magnolia, bietet CoreMedia nicht. Hochgeladene Bilder werden allerdings vom System automatisch kleiner skaliert. Bei den Bildern für das Carousel und in den Artikeln hat bei großen Auflösungen dies eine sehr verpixelte Darstellung zur Folge. Es ist daher wichtig bei der Redaktionierung der Bilder den Haken bei „Originalgröße verwenden“ zu setzen, um die volle Bildqualität präsentieren zu können.

Zusammenfassung der Anpassung am System:

- Navigation (Template) - CSS-Klasse
- Carousel (Template) - Kontrollpanel, Bewertung- und Kommentarfunktion
- Viewbaumstruktur Teaser/Artikel - separieren zwischen Teaser und Artikel
- Artikel (Template) -
  Anordnung Bild zu Text, Datum, Bewertung- und Kommentarfunktion, Überschrift
- Teaser (Template) -
  Linktext statisch, Datum, Bewertung- und Kommentarfunktion, Überschrift
- Footer (Template) - Copyrighttext
- Carousel (JavaScript) - Skalierung

4.4 Adobe Experience Manager 6

Bei der Umsetzung der prototypischen Beispielseite in Adobes AEM 6 wird als Grundlage, wie auch bei den anderen beiden CM Systemen, das mitgelieferte Standardtemplate genutzt. AEM 6 ermöglicht es, anders als Magnolia in der Community-Variante, eine Multimandantenkonfiguration zu nutzen. Die Umsetzung des Prototypen lässt sich ohne Eingriff in die Grundstruktur der Defaultseite realisieren, da man immer an einem Duplikat aller Templates und Einstellungen arbeitet.

Die Demoseite bei Adobe nennt sich „geometrixx“ und ist, wie bei den beiden CM Systemen zuvor, bereits mit Komponenten und Demoinhalt befüllt. Der Aufbau der Seite beginnt mit einem Kopfbereich, in dem zwei Links zur Personalisierung und eine Suche verankert sind. Die Navigationsleiste beinhaltet ein Logo, sowie alle redaktionierten Unterseiten. Der Hauptteil der Seite ist in zwei Bereiche aufgeteilt. Der erste Bereich ist mit einem statischen Bild gefüllt. Alle Texte und Buttons sind bereits im Bild integriert und keine dynamisch vom CM System erzeugten Inhalte. Das gesamte Bild beinhaltet eine Verlinkung auf eine andere Seite und ist somit in dieser Form für die Umsetzung des Prototypen, im Gegensatz zu dem zweiten Bereich der Seite, nicht zu gebrauchen. Dieser teilt die Seite in drei gleich große Spalten, welche für die Umsetzung der Teaserübersicht auf der prototypischen Beispielseite hilfreich ist.

Der Inhalt der drei Spalten spielt bei der Implementierung keine Rolle und kann aus diesem Grund entfernt werden. Der Footerbereich besteht aus drei Links „About us“, „Privacy Policy“ und „Terms of us“, sowie einen Copyright-Schriftzug. Die Links werden dynamisch generiert. Alle im Ordner „Footer“ liegenden Seiten werden in den Footerbereich eingebunden.

Das CRXDE bildet bei AEM 6 das Werkzeug für alle Frontendentwickler. In diesem vom Layout an den Windows Explorer angepassten Editor, lassen sich alle Templateänderungen direkt am System durchführen. In dieser Umgebung ist das

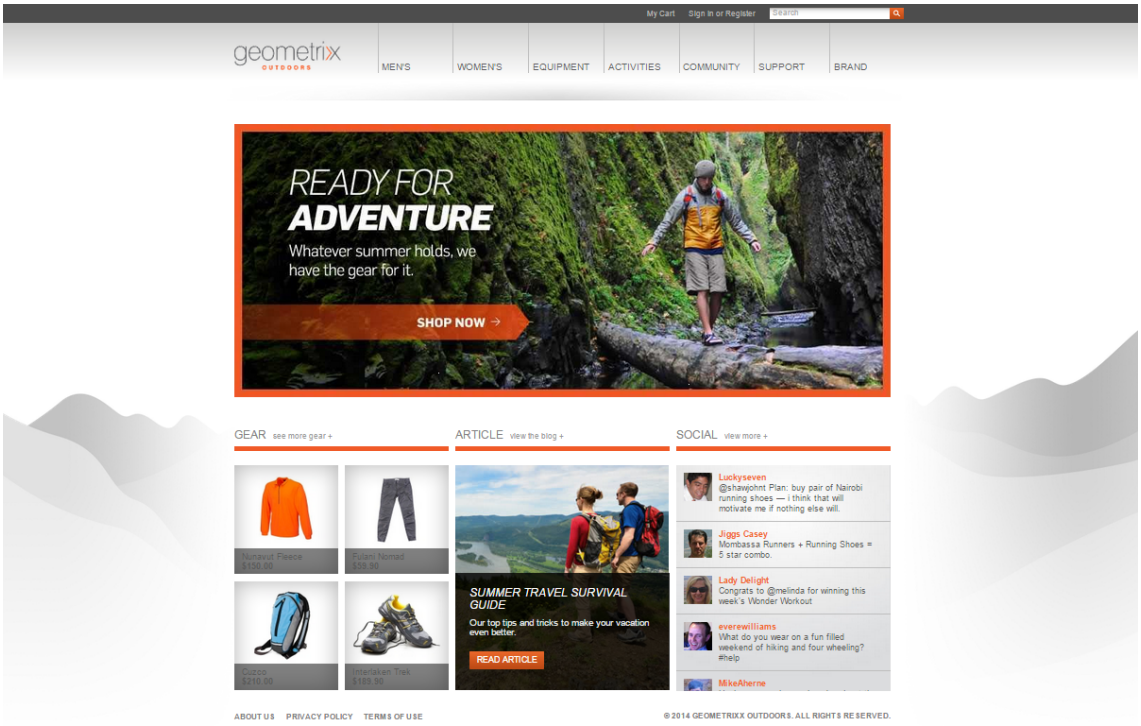

Abbildung 4.15: AEM 6, Demoseite

Einbinden von CSS- und JavaScript-Dateien, ebenso wie das Anpassen von Inhalten möglich. Um eine mandantenspezifische Anpassung der Templates, sowie des Layouts durchzuführen, wird der Demoprojekt-Ordner unter „app“ dupliziert und in „Diplomarbeit“ umbenannt. In diesem Ordner liegen alle, für die Grundstruktur der Seite benötigten, JSP-Dateien. Die einzelnen Komponententemplates liegen in einem gesonderten Pfad. Da die Templates der Grundstruktur bereits dupliziert wurden und somit ein Verweis auf die Komponenten in diesen verwaltet wird, können auf Seiten des Frontends auch mandantenspezifische Komponenten erstellt werden.

Nach erfolgreichem Duplizieren und Anpassen des Ordners unter „app“, muss dieser noch als Template im Contentbereich des Projektes eingetragen werden. Anschließend nutzt AEM 6 die JSP-Dateien aus dem neu erzeugten Ordner für das Rendern der Seite.

Die Projektseite ist im Template in drei Bereiche aufgeteilt: Header, Content und Footer. In der Header-JSP-Datei werden alle Elemente, bis auf die Navigationsleiste, entfernt und somit das Logo, die Suche und die Links aus der Darstellung des Headerbereiches ausgeschlossen. Ähnlich wie bei Magnolia und CoreMedia muss bei der Struktur der Navigation lediglich der CSS-Klassenname hinzugefügt werden, um das Plug-In nutzen zu können. Einzige Hürde bei der Anpassung der Navigation ist die Caching-Funktion von AEM 6. Um ein ständiges Generieren der Navigationsleiste zu vermeiden, wird der HTML-Abschnitt in einer Variable im System gespeichert. Diese Variable wird mit

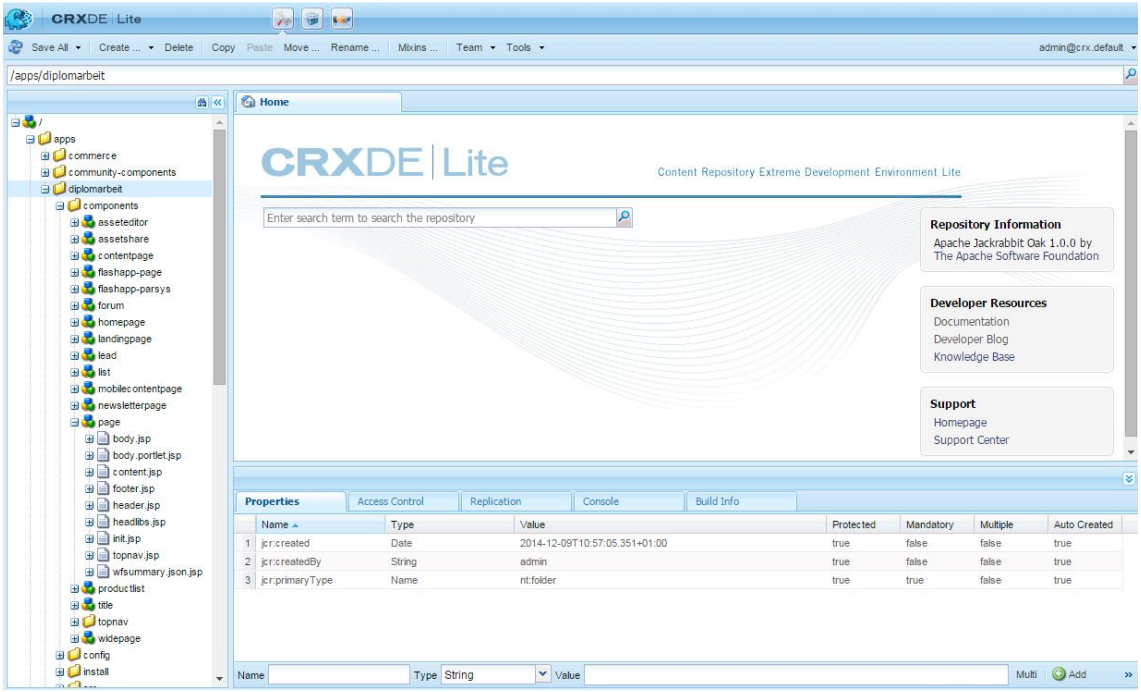

Abbildung 4.16: AEM 6, CRXDE

JavaScript ausgegeben. Um die Templateanpassung der Navigationsleiste testen zu können, erfordert es eine inhaltliche Anpassung eines der in der Leiste eingebunden Elementes.

Adobe liefert, wie auch die beiden Vergleichssysteme, eine Carousel Komponente. Bei AEM 6 gibt es eine Vielzahl von Auswahlmöglichkeiten, mit welchen Items das Carousel gefüllt werden soll. Für die prototypische Umsetzung in AEM 6, wird die „fixed list“ genutzt. Bei dieser Auswahl werden der Komponente redaktionell die Seiten hinzugefügt, welche jeweils ein Item des Carousels bilden. Es wurden für die Umsetzung direkt die drei in der Navigation angezeigten Seiten genutzt. Es ist selbstverständlich auch möglich, Seiten auszuwählen, welche nicht in der Navigationsleiste auftauchen sollen. Um die ausgewählten Seiten mit dem richtigen Inhalt anzeigen zu lassen, werden einige Anpassungen in den „Page Properties“ benötigt.

Für eine exakte Darstellung analog zum Prototypen, musste das Template der Carousel-Komponente ebenfalls noch angepasst werden. Auch bei AEM 6 war eine dynamische Anpassung des Linkbuttons nicht in der entsprechenden Komponente vorgesehen. Da eine Erweiterung des Dialoges der „Page Property“ eine Bearbeitung der Javaklasse nach sich zieht, wurde der „Page Titel“ für die Umsetzung dieses Buttons verwendet. Das Template des Carousel wurde um eine Variable „linkButton“ erweitert. Diese liest mit Hilfe der Getter-Methode „getPageTitle()“ den Inhalt des Feldes „Page Title“ aus. Somit kann der Button dynamisch redaktioniert werden. Der Seitentitel wurde

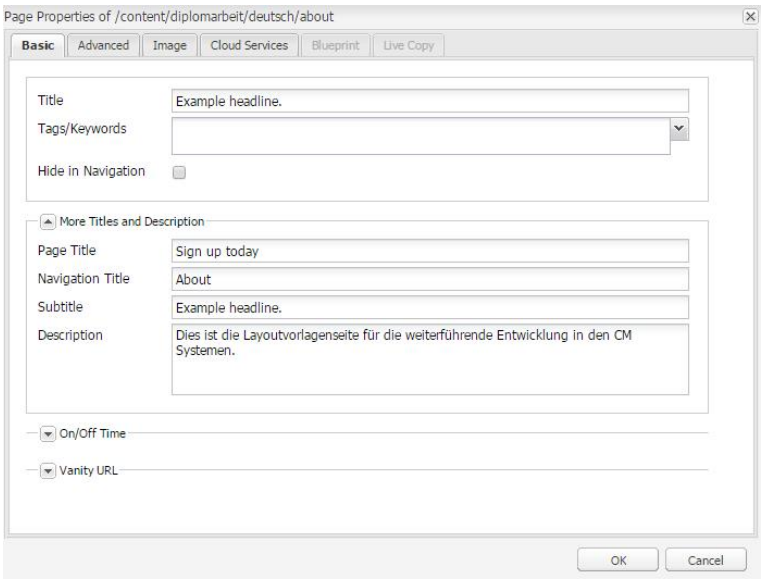

Abbildung 4.17: AEM 6, Page Properties

aus den anderen Templates entfernt und nur noch der „Title“ verwendet, um eine fehlerhafte Darstellung des Titels zu vermeiden.

Eine weitere intensivere Anpassung musste bei der Kontrollleiste der Komponente vorgenommen werden. AEM 6 verwendet hierbei Links, welche mit einem Hintergrundbild bestückt sind. In dem Linkelement befindet sich ein Dummybild, welches auf die Größe 25x25 Pixel skaliert wird. Der Mouse-Over-Effekt löst hierbei eine Positionsänderung des Bildes aus. Das bedeutet vereinfacht gesagt, dass der Link ein Bild als Hintergrundbild beinhaltet. Auf diesem ist sowohl der ausgefüllte, sowie der nur umrandete Kreis zu sehen. Durch den Positionssprung kann zwischen den beiden Bildern hin und her gewechselt werden. Somit scheint es, als würde sich der Kreis füllen, sobald man mit dem Mauszeiger darüber geht. Letztlich ändert sich lediglich das Bild und hat diesen optischen Effekt zur Folge.

Die Umsetzung der Artikel und Teaser auf der Seite unterscheiden sich bezüglich der Redaktionierung stark, im Vergleich zu den anderen beiden CM Systemen. Bei AEM 6 werden alle Teile einzeln eingebunden. Das bedeutet, dass bei den dreispaltigen Teasern nacheinander in die 3-Spalten-Komponente ein Bild, eine Überschrift, ein Text und ein Linktext eingefügt werden. Das ermöglicht zwar eine einfache Art die Inhalte zu bearbeiten, weist aber in der Folge Probleme bei der CSS-Gestaltung auf. Das folgende Beispiel erläutert das CSS-Problem genauer. Für den Linkbutton und den Text werden die selben Komponenten genutzt. Sie nutzen also die selben Templates und folglich bekommen sie auch die selben CSS-Klassen. Dies erschwert eine Unterscheidung bei der separaten Gestaltung der einzelnen Elemente. AEM 6 bietet hierbei auch nicht die Möglichkeit auf redaktionellem Weg den Komponenten CSS-Klassen mitzugeben. Dies wäre ein praktischer und softwareseitiger

Lösungsansatz, welcher das Gestaltungsproblem lösen kann.

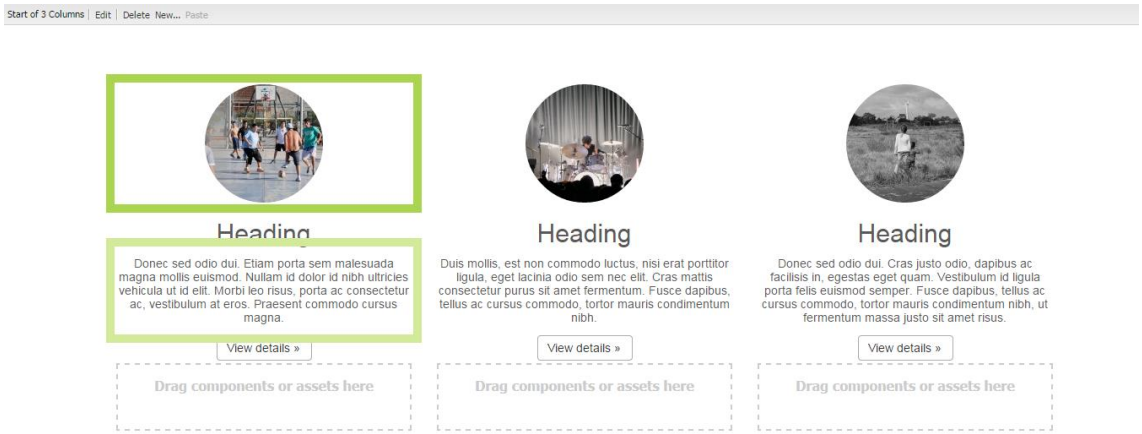

Abbildung 4.18: AEM 6, Edit Modus

Die selbe Problematik findet sich bei der Erstellung der Artikel wieder. Zum Einen gibt die einzelne Redaktionierung der Komponente dem Frontendentwickler die Möglichkeit die Reihenfolge der Bilder und Texte händig zu bearbeiten. Eine eindeutige Unterscheidung bei der Umsetzung des Layouts fehlt jedoch. Um diesen Problemen der fehlenden CSS-Klassen entgegen zu wirken, wird im Zuge dieser Diplomarbeit auf zwei Lösungsarten zurückgegriffen. Zum Ersten bietet CSS die Möglichkeit ein bestimmtes Kinderelement anzusprechen. Da der Aufbau der Teaser immer der Gleiche ist, kann diese Funktion für die Anpassung des Designs verwendet werden. Bei den drei Artikeln kommt die zweite Variante zum Einsatz. Hierbei wird mittels JavaScript, jedem zweiten Element die CSS-Klasse „right“ angehangen, somit kann ein alternierendes Design umgesetzt werden. Die Möglichkeit der einzelnen Redaktionierung bringt noch einen weiteren Vorteil mit sich. Es müssen für diese Seitenelemente keinerlei Templateanpassung vorgenommen werden, da alles im Edit-Modus der Seite realisiert werden kann.

Ein derartiger Eingriff bleibt einem bei der Anpassung des Fußbereiches nicht erspart. Es handelt sich bei den beiden Links im Prototypen um seitenunabhängige Links. Aus diesem Grund werden sie für diesen Mandanten fest ins Template implementiert. Es ist weiterhin möglich den Fußbereich mit Seiten aus dem „footer“-Ordner zu erweitern. Dies benötigt mit hoher Wahrscheinlichkeit eine anschließende Layoutanpassung. Der Copyright-Schriftzug wird ebenso direkt im Template angepasst.

Die maximale Breite der Navigationsleiste und der Schriftzug „Diplomarbeit“ als erstes Listenelement erfordern, wie bei den beiden Vergleichssystemen, eine JavaScript-Anpassung. AEM 6 markiert die aktuell besuchte Seite standardmäßig nicht in der Navigation mit einer CSS-Klasse. JavaScript wird auch hierfür als Lösungsmethode verwendet. Es erfolgt eine Prüfung des aktuell besuchten Pfades

und ein Vergleich mit den Zieladressen der Navigationspunkte. Sollte es zu einem Treffer kommen, wird dem gefundenen Navigationspunkt die CSS-Klasse „active“ zugewiesen.

Wie bereits in der Analyse von AEM 6 beschrieben, bietet Adobe die sogenannte Rendition Funktion. Diese führt bei der Umsetzung des Prototypen zu Problemen bei den Artikeln. Es ist beim Rendition nicht vorgesehen, Bilder auf die volle Breite des Bildschirmes zu skalieren. Die darüber liegende Zwei-Spalten-Komponente bestimmt, dass das Bild in den Maßen 270x270 Pixel ausgeliefert wird. Im maximalen Fall wird das Bild auf mobilen Endgeräten auf 940x940 Pixel gestreckt, was eine starke Verpixelung des Bildes nach sich zieht. Die Einstellung, welches Thumbnail gezogen wird, lässt sich nicht redaktionell im CMS anpassen. Somit bleibt das Bild in der mobilen Ansicht sehr verpixelt und bildet optisch eine starke Einschränkung.

Das Einbinden von CSS- und JavaScript-Dateien ist in der CRXCDE Umgebung sehr einfach händelbar und funktioniert problemlos. Die hervorgehobene Funktion, welche bei Magnolia CSS-Dateien in Abhängigkeit von Media Queries lädt, wäre ein tolles Feature bei der Strukturierung der Quelldateien. Ist jedoch mit einem passenden externen Programm zur Erstellung von CSS- und JavaScript-Dateien kein Hindernis bei der Kreierung von responsive Webseiten.

Zusammenfassung der Anpassung am System:

- Navigation (Template) - CSS-Klasse
- Carousel (Template) - Kontrollpanel, Linktext
- Footer (Template) - Copyrighttext

4.5 Ergebnis

In diesem Kapitel wurde die Implementierung der prototypischen Beispielseite, aufgeteilt auf die drei verschiedenen CM Systeme, beschrieben. Es erfolgte bei allen Systemen eine Beschreibung des Ausgangsinhaltes, des Aufbaus und des Grundlayouts. Anschließend wurden die einzelnen Implementierungsschritte beschrieben und Schwierigkeiten beziehungsweise praktische Features bei der Umsetzung des responsive Layouts näher erläutert.

Die erfolgreiche Integration der prototypischen Beispielseite in alle drei CMS bildet sogleich den Abschluss der praktischen Arbeit. Deren Kernaussage im nächsten Kapitel noch einmal zusammengefasst wird. Dafür erfolgt die endgültige Auswertung der zu Anfang aufgestellten Thesen, sowie eine kurze Bewertung der einzelnen CM Systeme bei der Umsetzung von responsive Design. Abgeschlossen wird die Diplomarbeit mit einem Fazit der gesamte Arbeit und einen kurzen thematischen Ausblick.

5 Zusammenfassung

5.1 Allgemeine Handlungsanweisungen

Ziel der Arbeit war es, Handlungsanweisungen zur Entwicklung von responsive Webseiten zu erstellen. Nach Abschluss der praktischen Arbeit konnten die folgenden fünf allgemeinen Handlungsanweisungen erstellt werden. Diese sollen einen Leitfaden bei der Entwicklung von responsive Design in CM Systemen bieten.

1. ***Entwicklung eines Prototypen***
   Es sollte ein Prototyp erstellt werden, welcher alle Funktionalitäten und Komponenten besitzt, die anschließend im CMS verwendet werden sollen. In den meisten Fällen werden mehrere Seiten mit verschiedenen Layouts erstellt. Im Prototypen sollte gekennzeichnet werden, welche Logos, Links und Texte dynamisch im System angepasst werden müssen.
2. ***Analyse des CM Systems***
   Mit Hilfe des Prototyps wird die Existenz und Funktionalität der erforderlichen Komponenten im CM System getestet. Anhand der Analyse erfolgt dann die Anpassung der Komponenten.
3. ***Anpassung oder Erstellung des responsive Verhaltens von speziellen Komponenten (zum Beispiel Carousel)***
   Manche Komponenten, welche mit Hilfe einer eigenen JavaScript-Datei funktionieren, arbeiten standardmäßig nicht responsive. Sollte bestimmten HTML-Tags im Code feste Größen oder Werte mitgegeben werden, welche nicht mit CSS überschrieben werden können, so muss die JavaScript-Datei der entsprechenden Komponente überarbeitet werden.
4. ***Einbindung responsive Features (zum Beispiel responsive Menü)***
   Sollten, wie in der Diplomarbeit, für die Umsetzung von responsive Menüs, besondere Plug-Ins benutzt werden, so werden diese nach dem Löschen des Grundlayouts in die Seite implementiert. Dabei ist die vorherige strukturelle Anpassung der Komponenten, welche in Punkt drei beschrieben wurde, wichtig.
5. ***Contentmigration und kundenspezifische Anpassungen***
   Nachdem alle Funktionen und Komponenten anhand des Prototypen entwickelt wurden, erfolgt die Redaktionierung der speziellen Seiten. Die für die Seite vorgesehenen Komponenten werden der Seite hinzugefügt und anschließend mit Inhalt gefüllt. Das gewünschte Layout wird zum Schluss mit Hilfe von CSS umgesetzt. Hierbei sollten bestenfalls keine Anpassungen mehr mit JavaScript erfolgen.

Wichtig ist, die im Theorieteil beschriebene, „mobile first“ Maxime. Bei der Umsetzung im vierten Kapitel dieser Diplomarbeit hat sich diese Herangehensweise bewährt. Je größer der Viewport nach erfolgreicher Anpassung des mobilen Layouts wird, desto geringer werden die Anpassungsaufwände an den einzelnen Komponenten. Anschließend müssen meist nur noch Schriftgrößen und Abmessungen, sowie Positionen angepasst werden.

5.2 Bewertung und Thesen

1. ***Responsive Design ermöglicht ein analoges Verhalten auf verschiedenen Endgeräten***
   Wie bereits mit der prototypischen Beispielseite bewiesen, ist es möglich mit responsive Design analoges Verhalten zu erzeugen. Alle Elemente und Bereiche der Seite passen sich hierbei prozentual an den Viewport der Seite an. Für eine optimale Darstellung wird jedoch das Neupositionieren von bestimmten Elementen empfohlen, um die Übersichtlichkeit zu bewahren. Damit ist diese These bewiesen.
2. ***Responsive Design unterstützt die Barrierefreiheit.***
   Responsive Design unterstützt, wie unter dem Punkt 2.1.2 gezeigt, die Erstellung von barrierefreien Webseiten.
3. ***Responsive Design Webseiten sind immer inhaltlich gekürzt.***
   Die Implementierung der prototypischen Beispielseite im Kapitel vier zeigt, dass eine Kürzung der Inhalte nicht notwendig ist. Eine verbesserte Anordnung der Elemente reicht aus, um die Seite nicht mit Inhalt zu überladen. Damit gilt diese These als widerlegt.
4. ***Alle Content Management Systeme brauchen ein responsive Design Feature.***
   Die Nutzung von Magnolia im vierten Abschnitt zeigte, dass responsive Design Features den Entwickler und Redakteur stark unterstützen können. Eine Implementierung der prototypischen Beispielseite konnte jedoch auch mit AEM 6 und CoreMedia realisiert werden, welche beide keine besonderen Features zur Umsetzung von responsive Design aufweisen. Folglich gilt auch diese These als widerlegt.
5. ***Nicht jedes Content Management System besitzt solch ein Feature.***
   AEM 6 und CoreMedia besitzen darüber hinaus beide keine prägnanten Features, welche bei der Umsetzung von responsive Design unterstützen können. AEM 6 bietet darüber hinaus die Möglichkeit im System den Previewmodus für die Betrachtung von responsive Design anzupassen. Diese These gilt aufgrund der fehlenden Feature bei CoreMedia als bewiesen.
6. ***Content Management Systeme sind für die Erstellung von responsive Design geeignet.***

Mit unterstützenden Drittprogrammen und einem schlüssigen Konzept ist eine Erstellung von responsive Design möglich. Das beweisen die CM Systeme CoreMedia und AEM 6, welche keine speziellen Umsetzungsunterstützungen aufweisen. Somit ist auch diese These bewiesen.

7. ***Allgemeine Handlungsanweisungen zur Umsetzung von responsive Websites unter Nutzung von verfügbaren Standardmechanismen bei Content Management Systemen sind erstellbar.***
   Die ähnliche Herangehensweise bei der Umsetzung der prototypischen Beispielseite im Kapitel vier, bei den drei ausgewählten CMS zeigten, dass der Ablauf der Umsetzung bei allen Systemen annähernd gleich war. Im Kapitel 5.1. wurden die allgemeinen Handlungsanweisungen definiert und somit gilt diese These als wahr.

5.3 Empfehlung für responsive Design

Als Abschluss dieser Diplomarbeit erfolgt eine kurze Zusammenfassung der drei untersuchten CM Systeme und eine Projektempfehlung. Wobei sich hier schon vorab sagen lässt, dass es mit keinen der drei Systeme unmöglich ist, responsive Seiten zu erstellen und zu verwalten.

Den Anfang macht AEM 6, welches durch seine vielen Anpassungsmöglichkeiten im Backend und mit der tollen Previewfunktion hervorsticht. Jegliche Frontendarbeiten waren im integrierten CRXDE sehr benutzerfreundlich zu lösen und aufgrund des bekannten „Windows-Explorer“ Layouts auch sehr intuitiv steuerbar. Das Rendition-Feature von AEM bedarf noch einer Backendanpassung, um eine Verwaltung der verschiedenen Größen direkt auf der Benutzeroberfläche möglich zu machen. Anschließend wäre dieses optimal zur Verbesserung der Performance von responsive Webseiten und würde eine ähnliche Funktionalität, wie das „Imaging“ bei Magnolia bieten. In Sachen JavaScript- und CSS-Dateien-Einbindung bietet das CRXDE eine ausreichend gute Oberfläche und stellt den Frontendentwickler vor keine Schwierigkeiten. Dasselbe lässt sich auch über die Templateanpassung sagen, welche ebenso gut im CRXDE zu bewältigen ist. AEM 6 zielt, wie auch schon seine Vorgänger, auf Großprojekte ab, wo eine umfangreiche Backendentwicklung eingebunden ist. AEM 6 setzt in Sachen Entwicklungsmöglichkeiten keine großen Grenzen mehr. Es sollte, nach Anpassungen des Backendes, möglich sein mit diesem Produkt eine detailgenaue Umsetzung der Kundenwünsche durchzuführen. Die umfangreiche Dokumentation sollte bei der Backendentwicklung sehr hilfreich sein.

CoreMedia reiht sich in Sachen Projektgröße in die selbe Kategorie, wie AEM 6. Aufgrund seines abweichenden Redaktionsprinzips ist es eher für große Datenbanken und Produktseiten geeignet, bei der eine zentrale Pflege der Inhalte gewünscht wird. Diese fügen sich anschließend automatisch in die speziell angepassten Seiten ein.

Die Frontendentwicklung stellte bei CoreMedia von allen drei CM Systemen den größten Aufwand dar. Da die Defaulttemplates in einer Bibliothek-Datei eingebaut und somit nicht direkt auf der Benutzeroberfläche editierbar sind, zog dies einige Neustarts des System nach sich, um die Templateänderungen wirksam zu machen. Laut Dokumentation des Herstellers soll es jedoch möglich sein benutzerspezifische Templates in die Bibliotheken einzubinden und diese direkt im CM System zu verwalten. Das Einbinden der CSS- und JavaScript-Dateien funktioniert bei CoreMedia, wie auch bei den anderen beiden CM Systemen, problemlos. CoreMedia sticht mit der Möglichkeit heraus, Unterseiten direkt über der Benutzeroberfläche mit weiteren CSS- und JavaScript-Dateien zu befüllen. Das hat zur Folge, dass man in CoreMedia seitenspezifische Layouts erstellen kann, die unabhängig von der Elternseite sind. Die responsive Inhaltszusammenstellung von CoreMedia ist ein tolles Feature zur Erstellung von nutzerspezifischen Webseiten. Aufgrund des thematischen Schwerpunktes auf das Design, wurde diese Möglichkeit von CoreMedia nicht tiefer betrachtet.

Nach Abschluss der Arbeit lässt sich festhalten, dass Magnolia die meisten standardmäßig integrierten responsive Design Features bietet. All diese Features wirken sehr durchdacht und helfen dem Redakteur und Frontendentwickler bei der Erstellung von responsive Design. Der im Kapitel vier erbrachte Nachweis der Analyseergebnisse hat aber auch gezeigt, dass die meisten Features nur sehr gute Unterstützer sind, für die Entwicklung jedoch keinesfalls notwendig. Das Theme-Model in Magnolia kann für umfangreiche Seiten ein tolles Werkzeug sein, bedarf aber eines gut durchdachten Konzeptes. Im Bezug auf Performanceoptimierung stellt Magnolia seine Konkurrenten in dieser Diplomarbeit in den Schatten. Das „Imaging“ ist eine revolutionäre Idee der Magnolia-Entwickler und könnte in Kürze auch bei anderen CM Systemen auf dem Markt zu finden sein. Magnolia 5 empfiehlt sich für mittelgroße Projekte mit einem kleinen Entwicklerkreis. In Magnolia 5 ist es möglich, so gut wie alle Änderungen direkt im CMS durchzuführen. So war das Addieren von Feldern im Dialog von Komponenten, sowie das Anpassen der Templates im System, ohne großen Aufwand realisierbar.

Allgemein empfiehlt sich bei allen drei CMS das Hinzufügen von CSS-Klassen-Feldern. Keines der CMS macht es direkt im System möglich, Komponenten mit CSS-Klassen zu versehen. Bei der Umsetzung des prototypischen Layouts stellte dies in allen drei Systemen das größte Problem dar.

5.4 Fazit und Ausblick

Das Ziel der Diplomarbeit, eine prototypische Beispielseite in allen drei CM Systeme zu integrieren, war erfolgreich. Jedes CMS wies hierbei verschiedene Schwierigkeiten, welche zum Teil mit geringem, zum Teil mit größerem Aufwand gelöst werden konnte, auf. Bei der Implementierung zeigten sich jedoch schon weiterführende Probleme in den

verschiedenen Konzepten der einzelnen Systeme. So waren Verweise in der Carousel Komponente, in einem System, auf einen Artikel gerichtet, in einem anderen CMS wurde auf eine ganze Seite referenziert. Diese tiefer in die Struktur reichenden Themen konnten bei dieser Diplomarbeit nicht bis ins Detail geprüft werden.

Erwähnt werden muss hierbei auch, dass es sich bei der Umsetzung in dieser Diplomarbeit nur um einen Prototypen handelt. Dieser wurde mit branchenüblichen Komponenten versehen, die häufig bei der Umsetzung von Marketing- und Informationsseiten genutzt werden. Es wurden Anpassungen in den CM Systemen vorgenommen, welche bei mehreren Mandanten in den entsprechenden CMS zu Fehler hätten führen können. In der Regel wurde immer auf diese Problematik hingewiesen und eine theoretische Lösung gegeben. Aufgrund des deutlich größeren Mehraufwandes, blieb es in diesen Fällen bei der Theorie und es wurde anhand eines Prototypen die allgemeine Funktionalität gezeigt.

Bei der Implementierung entstand ein immer wiederkehrendes Problem in allen CM Systemen, welches bereits mehrfach in den vorangegangenen Kapiteln erwähnt wurde. Die fehlende Möglichkeit CSS-Klassen hinzuzufügen. Keines der CM Systeme bot die Möglichkeit, über die Benutzeroberfläche, den Komponenten benutzerspezifische CSS-Klassen anzuhängen. Diese Funktionalität hätte die Implementierung der prototypischen Beispielseite stark vereinfacht.

Nach Abschluss der Diplomarbeit lässt sich feststellen, dass Content Management Systeme immer zur Entwicklung von responsive Webseiten geeignet sind. Der Markt steckt hier wahrscheinlich noch in den Anfängen und wird sich in den nächsten Jahren immer mehr an diese Thematik wagen und den Redakteuren, sowie Entwicklern weitere Möglichkeiten bieten um responsive Webseiten zu entwickeln.

Dasselbe gilt im Allgemeinen für das Thema responsive Design und responsive Content. Diese Thematik bietet noch viele weitere Möglichkeiten, dem Nutzer eine auf ihn angepasste Seite auszuliefern. Ein Aspekt könnte der gezeigte Ansatz in CoreMedia sein. Mit Hilfe von Tags und Cookies die Seite, je nach Interessen des Nutzers, zu füllen. Das bezieht sich auch auf die, im theoretischen Teil angesprochene, Barrierefreiheit. Auch in diesem Bereich bietet responsive Design noch viele Möglichkeiten. Ein weiterer Punkt kann die Ausdehnung der Ausspielmöglichkeiten auf nicht digitale Geräte sein. Somit könnten in CM Systemen beispielsweise Printmedien erstellt werden, welche nur noch für Zeitung und Internetseite verschiedene Layouts bekommen. Dasselbe ist auch in Sachen Produktvermarktung möglich. Somit könnten Bedienungsanleitungen oder Beipackzettel von Produkten lediglich mit verschiedenen Layouts bestückt werden.

Großes Potential bietet die Abhängigkeit von Datenraten bei mobilen Geräten. Hierbei werden unterschiedliche Inhalte in Abhängigkeit der anliegenden Datenrate geladen. Das hat zur Folge, dass Seiten bei guter Datenrate mit allen Bildern und Features

geladen werden, hingegen bei schwacher Datenrate auf Bilder und optische Effekte verzichtet werden kann. Wichtig ist abschließend zu erwähnen, dass sich das Thema responsive nicht auf die mobilen Endgeräte beschränkt. Die mobilen Endgeräte könnten in Zukunft nur noch einen kleinen Teil dieses Themengebietes abdecken.

Am Schluss bleibt festzuhalten, dass das Thema responsive eine tolle Möglichkeit bietet seine Produkte, Inhalte und Interessen der breiten Masse auf verschiedenen Medien zu präsentieren. Gleichzeitig ist die Entwicklung in Sachen responsive allgemein und in CM Systemen noch lange nicht am Ende ihrer Entwicklung und kann in den nächsten Jahren noch für einige Neuerungen sorgen.

Anhang A: Anhang auf beiliegender CD

1. **CSS-Dateien für Magnolia 5**
   magnolia_5_middle.css
   magnolia_5_small.css
   magnolia_5_style.css
   magnolia_5_super.css
   magnolia_5_superlarge.css
   magnolia_5_wide.css

2. **CSS-Datei für Coremedia 7**
   coremedia_7.css

3. **CSS-Datei für AEM 6**
   aem_6.css

4. **JavaScript-Datei für Magnolia 5**
   magnolia_5.js

5. **JavaScript-Datei für Coremedia 7**
   coremedia_7.js

6. **JavaScript-Datei für AEM 6**
   aem_6.js

7. **JavaScript-Datei für responsive Menü Plug-In**
   responsivemobilemenu.js

8. **Verzeichnis der Prüfschritte für barrierefreie Internetseiten**
   BITV.pdf

Literaturverzeichnis

[AEM01] Dokumentation AEM 6,
URL: http://docs.adobe.com/docs/en/aem/6-0.html,
Stand: 23.12.2014

[CMS01] Medien Community,
URL: http://www.mediencommunity.de/content/u7-medienneutrale-daten,
Stand: 12.09.2014

[CMS02] Wikipedia: Day Software,
URL: http://de.wikipedia.org/wiki/Day_Software,
Stand: 13.09.2014

[CMS03] Wikipedia: Coremedia CMS,
URL: http://en.wikipedia.org/wiki/CoreMedia_CMS,
Stand: 13.09.2014

[CMS04] Wikipedia: Magnolia CMS,
URL: http://en.wikipedia.org/wiki/Magnolia_(CMS),
Stand: 13.09.2014

[CMS05] Plug-In: responsive mobile Menü,
URL: http://responsivemobilemenu.com/en/,
Stand: 13.11.2014

[ME01] Wikipedia: Recherche,
URL: http://de.wikipedia.org/wiki/Recherche,
Stand: 14.11.2014

[ME02] Wikipedia: Modell CMS,
URL: http://de.wikipedia.org/wiki/Modell,
Stand: 14.11.2014

[PT01] responsive Design Sprungmarken,
URL: http://www.responsive-design.com/images/responsive-wireframe.png,
Stand: 25.12.2014

[RD101] Die 3 Säulen des Responsive Webdesign,
URL: http://www.konversionskraft.de/trends/die-3-saeulen-des-responsive-webdesign.html,
Stand: 25.08.2014

[RD102] Das ist doch gar nicht “responsive“! Ein Guide gegen Bullshit-Bingo in der Web-Entwicklung“,
URL: http://t3n.de/news/responsive-design-web-entwicklung-504906/2/,
Stand: 25.08.2014

[RD103] Tim Kadlec, Praxiswissen Responsive Webdesign, O’Reily,
September 2013

[RD104] “A LIST APART“, Responsive Web Design,
URL: www.alistapart.com/articles/responsive-web-design/,
Stand: 27.08.2014

[RD105] Der Workflow im Responsive Web Design – Prototyping & Co.,
URL: http://blog.kulturbanause.de/2013/06/workflow-responsive-web-design-prototyping/,
Stand: 30.08.2014

[RD106] Der BITV-Test - Verzeichnis der Prüfschritte,
URL: http://www.bitvtest.de/bitvtest/das_testverfahren_im_detail/pruefschritte.html,
Stand: 09.01.2015

[XD01] Dokumentation AEM 6,
URL: http://docs.adobe.com/docs/en/aem/6-0.html,
Zeitraum: 01.10.2014-19.01.2015

[XD02] Dokumentation Magnolia,
URL: https://documentation.magnolia-cms.com/display/DOCS/Magnolia+5+Documentation,
Zeitraum: 01.10.2014-19.01.2015

[XD03] Dokumentation CoreMedia Blueprint 7,
URL: https://documentation.coremedia.com/cm7/overview/,
Zeitraum: 01.10.2014-19.01.2015

[Q01] Wikipedia: Barrierefreies Internet,
URL: http://de.wikipedia.org/wiki/Barrierefreies_Internet,
Stand: 29.12.2014

[Q02] Wikipedia: CSS,
URL: http://de.wikipedia.org/wiki/Cascading_Style_Sheets,
Stand: 29.12.2014

[Q03] Wikipedia: Design,
URL: http://de.wikipedia.org/wiki/Design,
Stand: 29.12.2014

[Q04] Wikipedia: Desktop-Computer,
URL: http://de.wikipedia.org/wiki/Desktop-Computer,
Stand: 29.12.2014

[Q05] Wikipedia: Blickfang,
URL: http://de.wikipedia.org/wiki/Blickfang,
Stand: 29.12.2014

[Q06] Wikipedia: Framework,
URL: http://de.wikipedia.org/wiki/Framework,
Stand: 29.12.2014

[Q07] Wikipedia: Zugriffsfunktion,
URL: http://de.wikipedia.org/wiki/Zugriffsfunktion,
Stand: 29.12.2014

[Q08] Wikipedia: JavaScript,
URL: http://de.wikipedia.org/wiki/JavaScript,
Stand: 29.12.2014

[Q09] Wikipedia: JavaServer Pages,
URL: http://de.wikipedia.org/wiki/JavaServer_Pages,
Stand: 29.12.2014

[Q10] Wikipedia: Klasse Programmierung,
URL: http://de.wikipedia.org/wiki/Klasse_%28Programmierung%29,
Stand: 29.12.2014

[Q11] Wikipedia: Layout,
URL: http://de.wikipedia.org/wiki/Layout,
Stand: 29.12.2014

[Q12] Wikipedia: Hyperlink,
URL: http://de.wikipedia.org/wiki/Hyperlink,
Stand: 29.12.2014

[Q13] Limeflavour: metanavigation,
URL: http://www.limeflavour.com/agentur/glossar/glossareintrag/metanavigation/,
Stand: 29.12.2014

[Q14] Wikipedia: Objekt (Programmierung),
URL: http://de.wikipedia.org/wiki/Objekt_(Programmierung),
Stand: 29.12.2014

[Q15] Wikipedia: Personalisierung (Informationstechnik),
URL: http://de.wikipedia.org/wiki/Personalisierung_(Informationstechnik),
Stand: 29.12.2014

[Q16] Wikipedia: Repository,
URL: http://de.wikipedia.org/wiki/Repository,
Stand: 29.12.2014

[Q17] Wikipedia: Smartphone,
URL: http://de.wikipedia.org/wiki/Smartphone,
Stand: 29.12.2014

[Q18] Wikipedia: Tabletcomputer,
URL: http://de.wikipedia.org/wiki/Tabletcomputer,
Stand: 29.12.2014

[Q19] Wikipedia: Teaser,
URL: http://de.wikipedia.org/wiki/Teaser,
Stand: 29.12.2014

[Q20] Wikipedia: Template Programmierung,
URL: http://de.wikipedia.org/wiki/Template_%28Programmierung%29,
Stand: 29.12.2014

[Q21] Wikipedia: Skin,
URL: http://de.wikipedia.org/wiki/Skin_(Computer),
Stand: 29.12.2014

[Q22] Wikipedia: Vorschaubild,
URL: http://de.wikipedia.org/wiki/Vorschaubild,
Stand: 29.12.2014

[Q23] Wikipedia: URL,
URL: http://de.wikipedia.org/wiki/Uniform_Resource_Locator,
Stand: 29.12.2014

[Q24] Wikipedia: Viewport,
URL: http://de.wikipedia.org/wiki/Viewport,
Stand: 29.12.2014

[Q25] Wikipedia: XML,
URL: http://de.wikipedia.org/wiki/Extensible_Markup_Language,
Stand: 29.12.2014

Glossar

Accordion nennt man in CMS die Komponenten, welche Inhalte in verschiedene aufklappbare Bereiche unterteilt.

Adobe Experience Manager (AEM) Content Management System der Firma Adobe Systems, trug ehemals den Namen „Day“, Nachfolgersystem von Adobe Communique (Adobe CQ).

Barrierefreiheit Barrierefreies Internet sind Web-Angebote, die von allen Nutzern unabhängig von ihren Einschränkungen oder technischen Möglichkeiten uneingeschränkt (barrierefrei) genutzt werden können [Q01].

Bootstrap ist ein freies und sehr häufig verwendetes CSS-Framework.

Breakpoint dt.: Sprungmarke, bezeichnet, in dieser Diplomarbeit, einen definierten Punkt bei dem sich das Layout verändert.

button ist ein HTML-Tag, welches einen Button erzeugt.

Carousel nennt man eine Komponente, bei der, wie in einem Karussell, Bilder von links nach rechts durchs Bild laufen.

Cascading Style Sheets (CSS3) ist eine Gestaltungssprache für elektronische Dokumente und zusammen mit HTML und DOM, eine der Kernsprachen des World Wide Webs. CSS3 ist die aktuelle Version der CSS Reihe [Q02].

Content Inhalte, z.B. eines Content Management Systems, eines Document Management Systems oder eines Wikis.

Content Management Systeme (CMS) sind Systeme zur gemeinschaftlichen Erstellung, Verarbeitung und Verwaltung von Inhalten und Workflows, gebräuchlich vor allem zur Erstellung von Webauftritten für technisch weniger versierte Nutzer.

CoreMedia Enterprise Content Management System der Firma Coremedia AG.

CRXDE nennt sich das in AEM integrierte Repository.

Customlayout bezeichnet ein benutzerspezifisch angepasstes Layout, welches vom Standardlayout abweicht.

Design dt: Gestaltung, bedeutet meist Entwurf oder Formgebung [Q03].

Deskotoprechner ist ein Computer in einer Gehäuseform, passend für den Einsatz als Arbeitsplatzrechner auf Schreibtischen [Q04].

Dialog In dieser Diplomarbeit: Ein Fenster in welchen alle nötigen Einstellungen einer Komponente eingetragen werden.

div ist ein HTML-Tag, welches einen Bereich auf der Seite definiert.

Document Management System (DMS) ist ein System zur gemeinschaftlichen Erstellung, Verarbeitung und Verwaltung von Dokumenten.

Dummybild Ein 1 Pixel großes leeres Bild. Wird genutzt um Zwischenräume und Klickbereiche für Links zu generieren.

Dummylink bezeichnet einen Link, der sich wie ein normaler Link verhält, jedoch kein eingetragenes Ziel hat.

Edit-Modus Bei den meisten CMS kann man direkt auf einer Seite Elemente bearbeiten und verwalten. Um dies zu realisieren wechselt man zwischen einen Edit, wo alle Bearbeitungen durchgeführt werden und einen Previewmodus. Hier wird die Seite exakt wie im Internet dargestellt.

Element ist ein Teil einer Menge und muss immer einer bestimmten Sache untergeordnet sein.

Enterprise Content Management System (ECMS) ist insbesondere für die Verwendung in großen Firmen mit viel Content und mehreren Ausspielkanälen (unter anderem Internet, Intranet) geeignet.

Extensible Markup Language (XML) Plattform- und implementierungsunabhängige Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten (Textdaten) [Q25].

Eyecatcher dt.: Blickfang, ist ein Grafik- oder Text-Element, das die Aufmerksamkeit des Betrachters auf eine bestimmte Botschaft lenken soll [Q05].

Footerbereich dt.: Fußzeile, ist der untere Teil einer Webseite, welche auf allen Unterseiten gleich dargestellt wird.

Framework ist ein Programmiergerüst, dass in der Softwaretechnik, insbesondere im Rahmen der objektorientierten Softwareentwicklung sowie bei komponentenbasierten Entwicklungsansätzen, verwendet wird [Q06].

Getter-Methode ist in der objektorientierten Programmierung eine spezielle Methode, die eine einzelne Eigenschaft eines Objekts abfragt [Q07].

Grid nennt man die einzelnen Spalten/Abschnitte auf einer Webseite, welche mittels verschachtelter „div“ Element kreiert wird.

Headerbereich dt.: Kopfzeile, ist der oberste Teil einer Webseite, in diesen werden die Navigation, Metanavigation und Titelbilder dargestellt.

Hypertext Markup Language 5 (HTML5) ist eine Auszeichnungssprache zur Strukturierung von Inhalten wie Text, Bildern und Hyperlinks in Dokumenten, Grundlage des World Wide Webs.

Item ist ein Punkt in einer Aufzählung.

JavaScript ist eine Skriptsprache, die für dynamisches HTML in Webbrowsern entwickelt wurde [Q08].

JavaServer Pages (JSP) ist eine von Sun Microsystems entwickelte, auf JHTML basierende Web-Programmiersprache zur einfachen dynamischen Erzeugung von HTML- und XML-Ausgaben eines Webservers [Q09].

Klasse versteht man in der objektorientierten Programmierung ein abstraktes Modell beziehungsweise einen Bauplan für eine Reihe von ähnlichen Objekten [Q10].

Kontrollelement bezeichnet in dieser Diplomarbeit das Steuerungspanel der Carousel Komponente, um zwischen den verschiedenen Bildern zu wechseln.

Konzept Auf Schema-Ebene einer Ontologie die kleinste Einheit (auf Daten-Ebene als „Entität“ bezeichnet), Konzepte können voneinander erben und durch Relationen verbunden sein.

Laptop ist ein Computer in einem tragbaren Gehäuse, welcher für den mobilen Einsatz genutzt werden kann.

Layout bezeichnet das detaillierte Sichtbarmachen eines gedanklichen Bildes im Sinne eines tatsächlichen Entwurfs, meist dem einer Drucksache [Q11].

Link Kurzform von Hyperlink, ist ein Querverweis in einem Hypertext, der funktional einen Sprung zu einem anderen elektronischen Dokument oder an eine andere Stelle innerhalb eines Dokuments ermöglicht [Q12].

Magnolia ist ein Content Management System der Firma Magnolia International, es ist in einer Community und Enterprise Variante erhältlich.

Mandant In dieser Diplomarbeit: Bezeichnung für einen einzelnen Webauftritt eines Kunden im CMS (beispielsweise ein Internet, ein Intranet oder ähnliches).

Media Queries fragen die Eigenschaften des Endgerätes direkt aus. Kann mittels CSS3 verschiedene Inhalte in Abhängigkeit der Eigenschaften des Endgerätes einbinden.

Metanavigation bezeichnet den Navigationsbereich im Header oder Footer einer Website und führt Links zum Beispiel zu Impressum, Richtlinien zum Datenschutz, Kontakt oder Sprachwechsel [Q13].

Mouse-Over-Effekt ist das besondere Verhalten von HTML-Elementen, wenn der Mauszeiger über das entsprechende Element gehalten wird.

Navigation beinhaltet alle Links, Menübeschriftungen und andere Elemente, die den Zugriff auf Webseiten ermöglichen und dem Benutzer bei der Orientierung innerhalb einer Website helfen.

Objekt bezeichnet in der objektorientierten Programmierung ein Exemplar eines bestimmten Datentyps oder einer bestimmten Klasse [Q14].

Open-Source Auch: quelloffen, Lizenzsammlung für Software, deren Quelltext frei zugänglich ist, in jeglicher Form weiterverbreitbar, auch kommerziell nutzbar.

Personalisierung bezeichnet die nominelle Zuordnung von Merkmalen zu einer nutzenden Person und die Anpassung von Programmen, Diensten oder Informationen an die persönlichen Vorlieben, Bedürfnisse und Fähigkeiten eines Benutzers [Q15].

Plug-in Optionales Erweiterungsmodul für ein Softwareprodukt, zur Erweiterung des Funktionsumfangs.

Portable Document Format (PDF) Plattformunabhängiges Format für Dokumente, entwickelt von der Firma Adobe Systems.

Preview Bei den meisten CMS kann man direkt auf einer Seite Elemente bearbeiten und verwalten. Um dies zu realisieren wechselt man zwischen einen Edit, wo alle Bearbeitungen durchgeführt werden und einen Previewmodus. Hier wird die Seite exakt wie im Internet dargestellt.

Publisher bezeichnet man den Webserver, welcher im Internet allen Inhalt zur Darstellung besitzt. Aktiviert man in einem CMS eine Seite, so wird diese auf den Publisher geschoben und ist somit im Internet aktiv.

Rendern nennt man die dynamische Erzeugung von HTML- und XML-Ausgaben eines Webservers.

Rendition ist eine besondere Funktion von AEM 6, um Bildergrößen auf Objekttypen anzupassen.

Repository ist ein verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten [Q16].

responsive dt.; reagierend, Der Inhalt beziehungsweise dessen Darstellung kann sich variabel auf diverse Endgeräte und Auslieferungsmodi einstellen.

Sliderbox Eine Box, welche Slides/Folien beinhaltet.

Smartphone ist eine Weiterentwicklung des Mobiltelefons, welches mehr Computer-Funktionalität und -konnektivität als ein herkömmliches fortschrittliches Mobiltelefon zur Verfügung stellt [Q17].

span ist ein HTML-Tag, welches Inhalt gruppiert und keine Layoutänderungen nach sich zieht.

Standardmechanismen sind die in einem System vorhanden Funktionen, welche keine Anpassung Dritter beinhaltet.

Tablet ist ein tragbarer, flacher Computer in besonders leichter Ausführung mit einem Touchscreen, aber, anders als bei Notebooks, ohne ausklappbare mechanische Tastatur [Q18].

Teaser ist in der Werbe- und Journalismussprache ein kurzes Text- oder Bildelement, das zum Weiterlesen, -hören, -sehen, -klicken verleiten soll [Q19].

Template ist eine Mustervorlage oder Schablone für ein Dokument, dass die wesentlichen Layout-Elemente enthält und mit Grafiken und Texten gefüllt werden kann [Q20].

Theme ist eine Zusammenstellung von grafischen Elementen und Einstellungen, die das Aussehen und Verhalten einer Websites festlegen [Q21].

Thumbnails werden kleine digitale Grafiken oder Bilder bezeichnet, die als Vorschau für eine größere Version dienen [Q22].

ul ist ein HTML-Tag, welches eine Liste ohne Aufzählungszeichen erzeugt.

Unified Resource Locator (URL) ist eine Unterart von URIs, dient der Identifikation und Lokalisierung einer Ressource über eine Zugriffsmethode.

Variation ist das Unterscheidungsmodell der verschiedenen Endgeräte bei Magnolia.

Viewport ist der Anzeigebereich im Fenster einer Anwendung, der für die Darstellung des Anwendungsinhaltes tatsächlich zur Verfügung steht.

Web Content Management System (WCMS) ist ein Content Management System, dient speziell der Ausspielung der Inhalte im Web.

Workflow Inhaltlich abgeschlossene, zeitlich und logisch zusammenhängende Funktionsfolge.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 21.01.2015